L Number	Hits	Search Text	DB	Time stamp
-	140	ajanovic	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/21 14:48
-	100	interface and ajanovic	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/21 15:24
-	23695	(access request) near3 type	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/21 15:25
-	138148	host bridge and ((access request) near3 type)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/21 15:30
-	137	"host bridge" and ((access request) near3 type)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/21 15:39
-	5319	"bus bridge" "host bridge"	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/22 09:54
-	23695	(access request) near3 type	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/22 09:55
-	663032	critical "real-time" "real time" realtime isochronous "best effort"	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/22 09:56
-	1693	("bus bridge" "host bridge") and (critical "real-time" "real time" realtime isochronous "best effort")	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/22 09:57
-	216	(("bus bridge" "host bridge") and (critical "real-time" "real time" realtime isochronous "best effort")) and ((access request) near3 type)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/22 09:58
-	334648	arbit$ (((("bus bridge" "host bridge") and (critical "real-time" "real time" realtime isochronous "best effort")) and ((access request) near3 type))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/22 09:58
-	175	arbit$ and (((("bus bridge" "host bridge") and (critical "real-time" "real time" realtime isochronous "best effort")) and ((access request) near3 type))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/22 10:47
-	796	("bus bridge" "host bridge") and critical	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/22 10:47

US006105094A

United States Patent [19]

Lindeman

[11] Patent Number: 6,105,094

[45] Date of Patent: *Aug. 15, 2000

[54] **METHOD AND APPARATUS FOR ALLOCATING EXCLUSIVE SHARED RESOURCE REQUESTS IN A COMPUTER SYSTEM**

[75] Inventor: **James R. Lindeman**, Fremont, Calif.

[73] Assignee: **Adaptec, Inc.**, Milpitas, Calif.

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **09/013,498**

[22] Filed: **Jan. 26, 1998**

[51] **Int. Cl.**[7] **G06F 13/14**

[52] **U.S. Cl.** **710/107**; 710/108; 710/240; 710/241

[58] **Field of Search** 710/240, 241, 710/242, 243, 244, 107, 108, 109, 110

[56] **References Cited**

U.S. PATENT DOCUMENTS

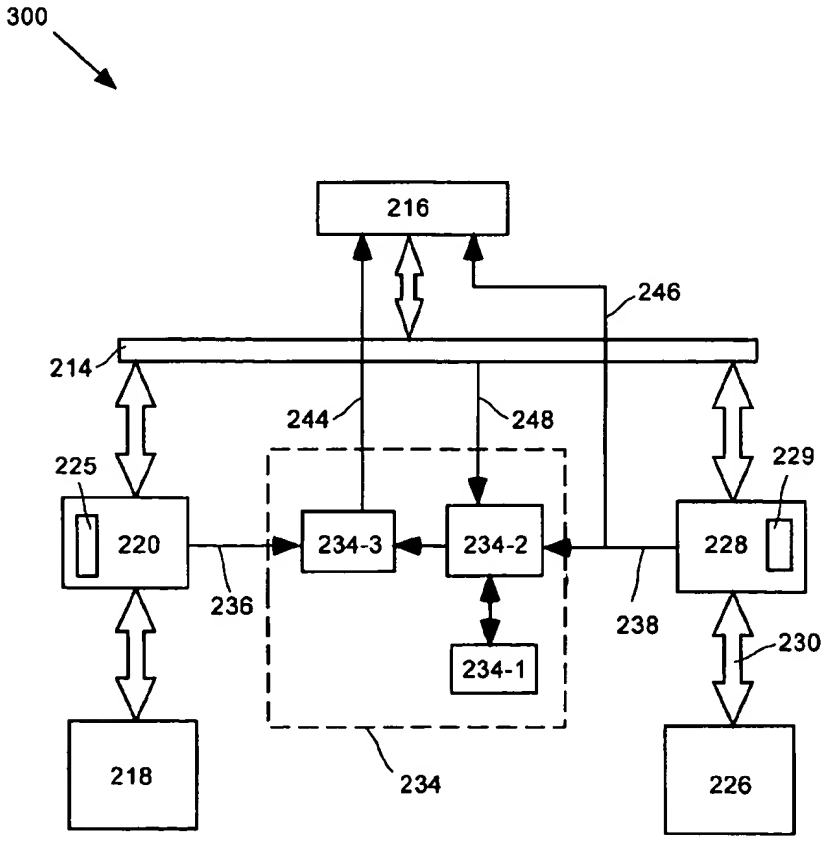
4,719,569	1/1988	Ludemann et al.	364/200
5,528,766	6/1996	Ziegler et al.	395/293
5,533,205	7/1996	Bickledge, Jr. et al.	395/297
5,546,547	8/1996	Bowes et al.	395/294
5,640,519	6/1997	Langendorf et al.	395/291
5,758,105	5/1998	Kelly et al.	395/293

Primary Examiner—Glenn A. Auve
Assistant Examiner—Tim Vo
Attorney, Agent, or Firm—Martine Penilla & Kim, LLP

[57] **ABSTRACT**

A computer system incorporating an apparatus for allocating exclusive shared resource requests includes a shared resource, a first type device coupled to the shared resource and a second type device coupled to the shared resource. The computer system also includes an arbitrator unit coupled to the shared resource capable of granting the second type device exclusive access to the shared resource by preventing the first type device from being granted exclusive access to the shared resource. The arbitrator unit prevents the first type device from being granted exclusive access to the shared resource for at least a duration of time after the second type device has completed an associated second type shared resource transaction.

19 Claims, 6 Drawing Sheets



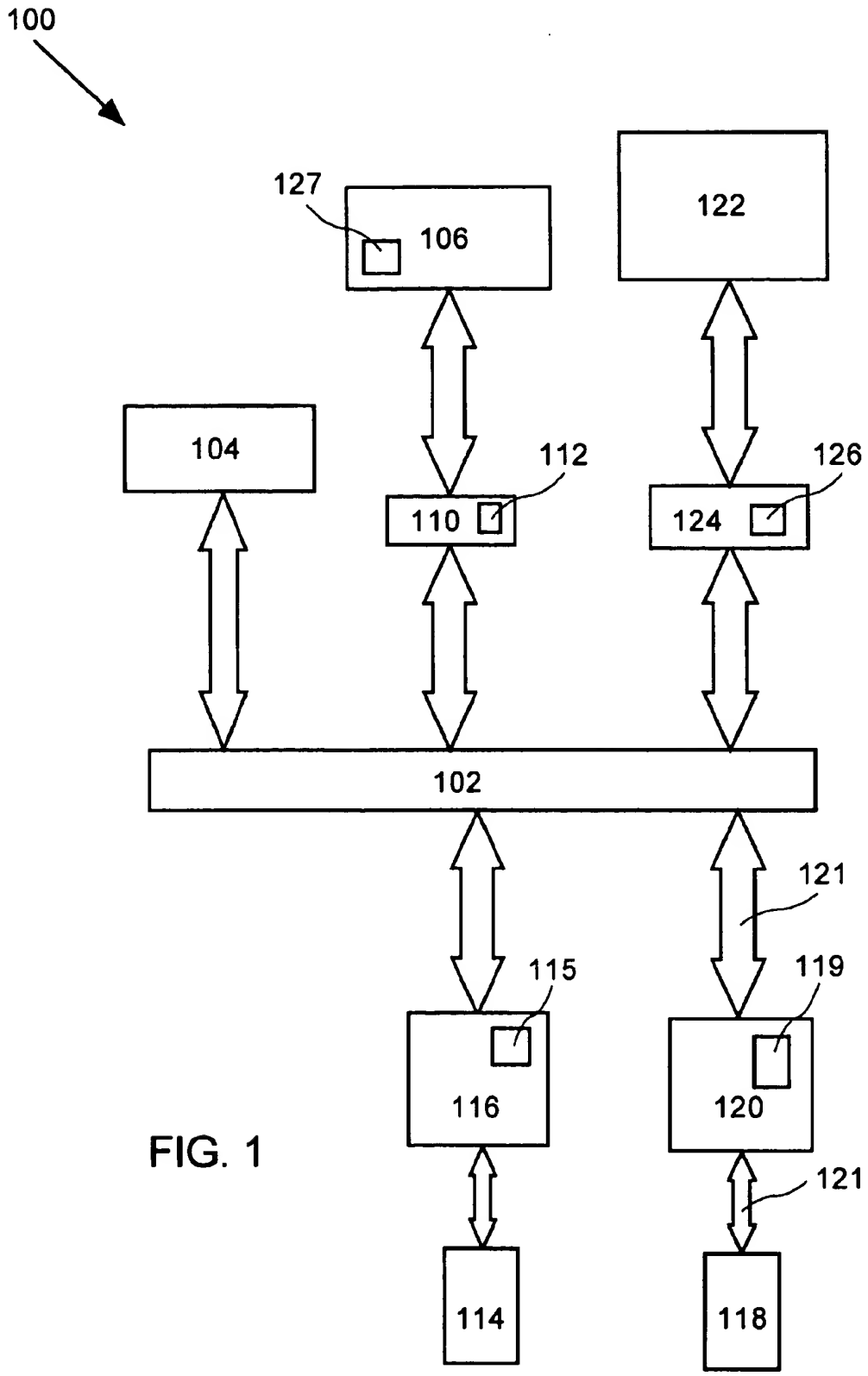

100
127
106
122
104
112
126
110
124
102
121
115
119
116
120
121
114
118

FIG. 1

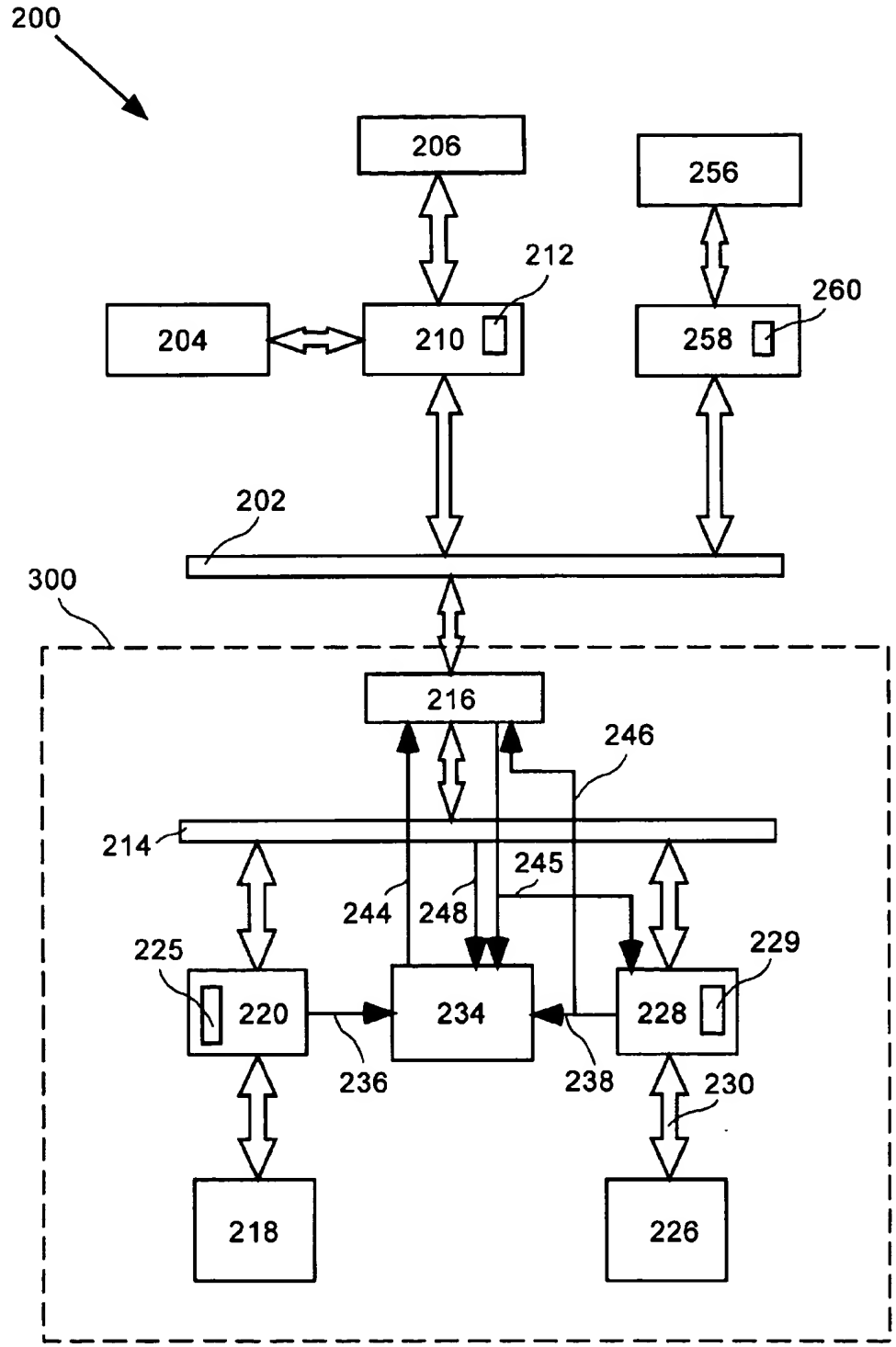

200
206
256
212
260
204
210
258
202
300
216
246
214
245
244
248
225
229
220
234
228
236
238
230
218
226

FIG. 2

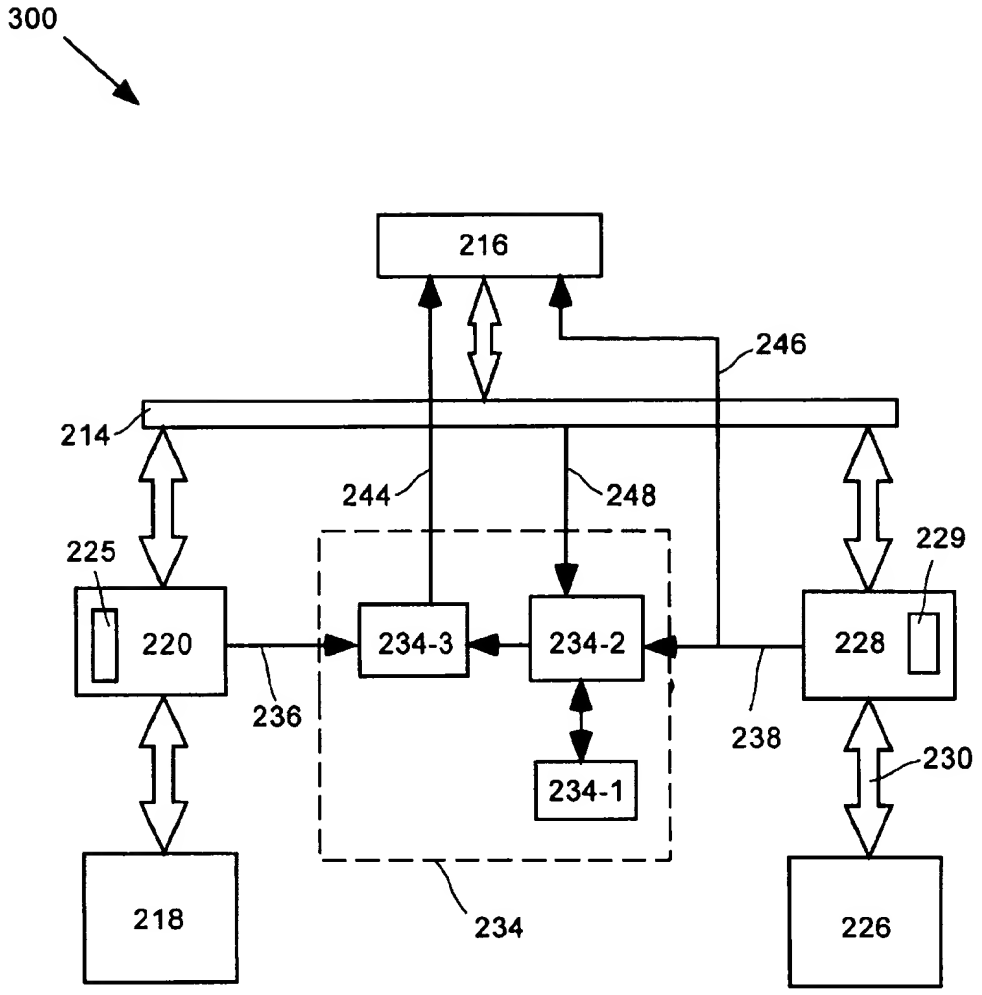

300
216
246
214
244
248
225
229
220
234-3
234-2
228
236
238
230
234-1
218
234
226

FIG. 3

US005787264A

United States Patent [19]

Christiansen et al.

[11] Patent Number: 5,787,264

[45] Date of Patent: Jul. 28, 1998

[54] METHOD AND APPARATUS FOR ARBITRATING ACCESS TO A SHARED BUS

[75] Inventors: Kevin M. Christiansen, Saratoga; Mark A. Stubbs, Felton; Bruce Eckstein, Cupertino, all of Calif.

[73] Assignee: Apple Computer, Inc., Cupertino, Calif.

[21] Appl. No.: 437,233

[22] Filed: May 8, 1995

[51] Int. Cl.[6] H01J 13/00

[52] U.S. Cl. 395/293; 395/288; 395/299; 395/290; 395/726; 395/297

[58] Field of Search 395/290, 293, 395/296, 297, 299, 300, 304, 800, 291, 845, 298, 305, 726, 727, 729, 731, 732, 303

[56] References Cited

U.S. PATENT DOCUMENTS

4,987,529	1/1991	Craft et al.	395/293
5,237,696	8/1993	Best	395/293
5,301,283	4/1994	Thacker et al.	395/293
5,392,436	2/1995	Jansen et al.	395/293
5,438,666	8/1995	Craft et al.	395/842
5,506,989	4/1996	Boldt et al.	395/732
5,519,838	5/1996	Ziegler et al.	395/299
5,528,767	6/1996	Chen	395/293
5,572,686	11/1996	Nunziata et al.	395/296
5,581,782	12/1996	Sarangdhar et al.	395/800

Primary Examiner—Ayaz R. Sheikh
Assistant Examiner—Raymond N. Phan
Attorney, Agent, or Firm—Burns, Doane, Swecker & Mathis, L.L.P.

[57] ABSTRACT

The present invention is directed to providing a computer system which arbitrates control of a shared bus among plural devices included in the computer system. In accordance with the present invention, at least one of the devices is afforded a higher priority than the remaining devices, yet none of the remaining devices are effectively denied system bus access or control for extended periods of time. The present invention can therefore increase operating efficiency even as the number of devices included in the computer system is increased to achieve enhanced processing power. In addition, the present invention can provide sophisticated multimedia features, including real time signal processing, without sacrificing overall operating efficiency. In accordance with the present invention, the plural devices arbitrate system bus control in a manner which achieves acceptable multimedia results when processing real time data streams such as video data streams, audio data streams, animation data streams, and so forth, yet which does not sacrifice the access of remaining devices in the computer system to the shared bus.

20 Claims, 3 Drawing Sheets

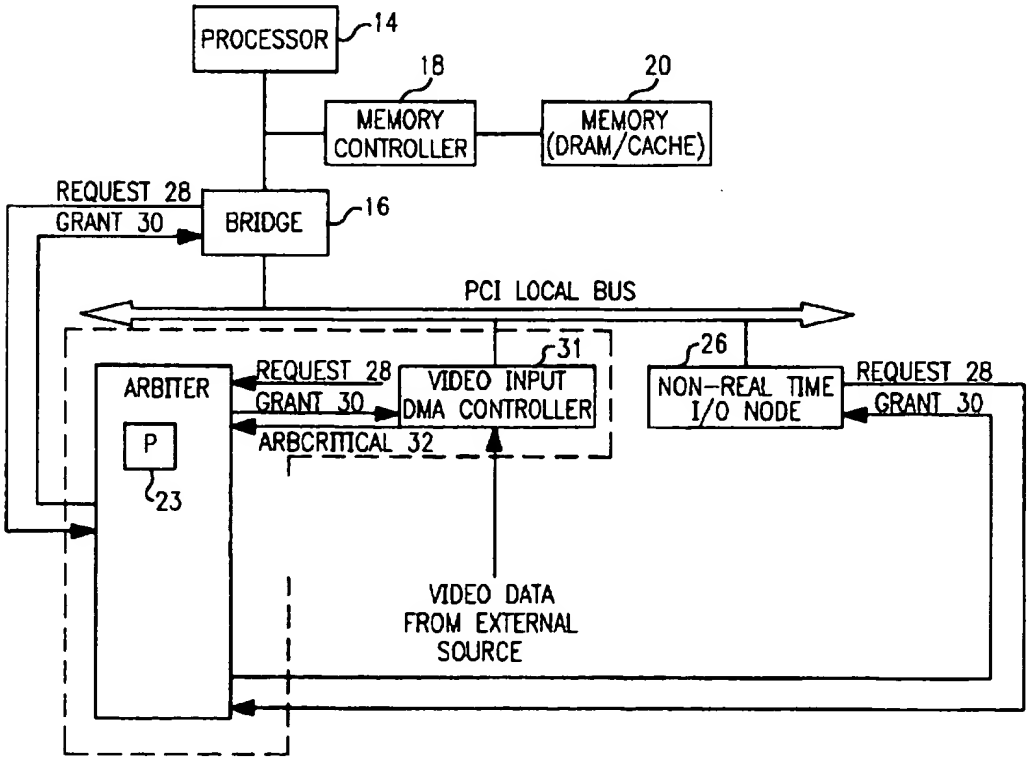


FIG. 1

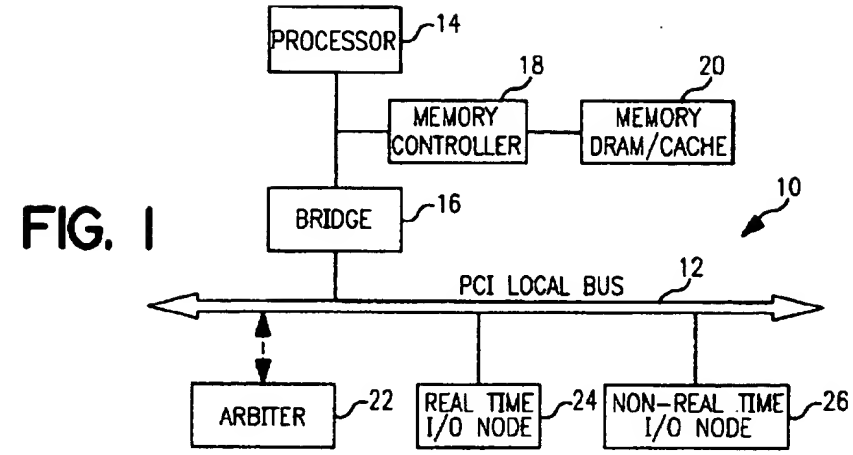
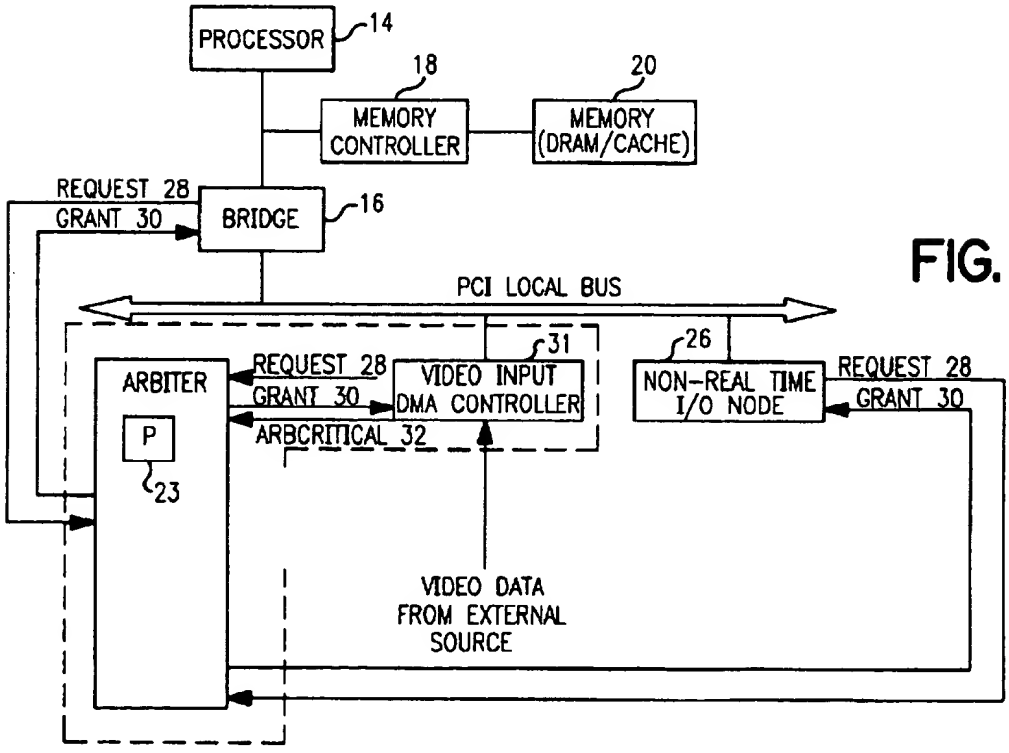

PROCESSOR
14
18
20
MEMORY CONTROLLER
MEMORY DRAM/CACHE
BRIDGE
16
10
PCI LOCAL BUS
12
ARBITER
22
REAL TIME I/O NODE
24
NON-REAL TIME I/O NODE
26

FIG. 2


PROCESSOR
14
18
20
MEMORY CONTROLLER
MEMORY (DRAM/CACHE)
REQUEST 28
GRANT 30
BRIDGE
16
PCI LOCAL BUS
31
26
ARBITER
REQUEST 28
GRANT 30
VIDEO INPUT DMA CONTROLLER
NON-REAL TIME I/O NODE
REQUEST 28
GRANT 30
ARBCRITICAL 32
P
23
VIDEO DATA FROM EXTERNAL SOURCE

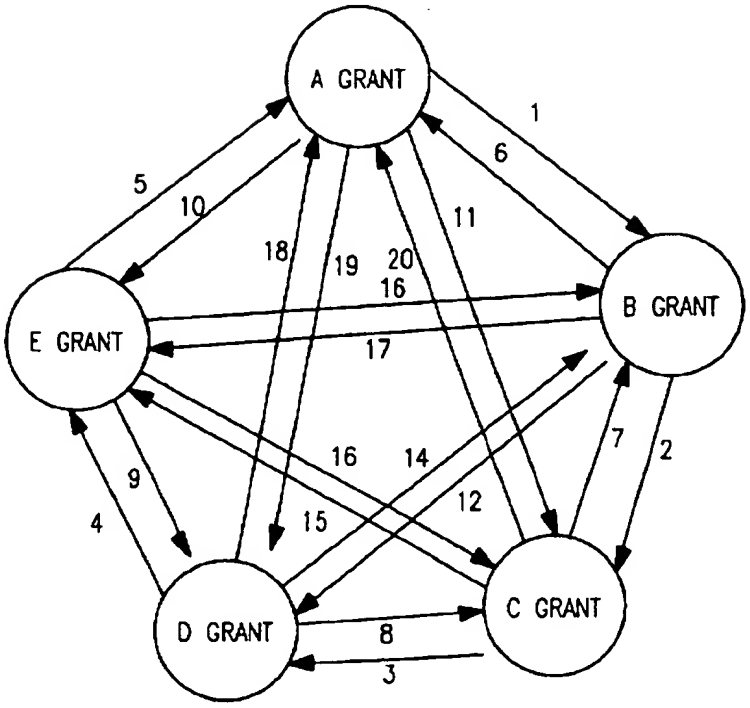

A GRANT
B GRANT
C GRANT
D GRANT
E GRANT
1
2
3
4
5
6
7
8
9
10
11
12
14
15
16
16
17
18
19
20

FIG. 3

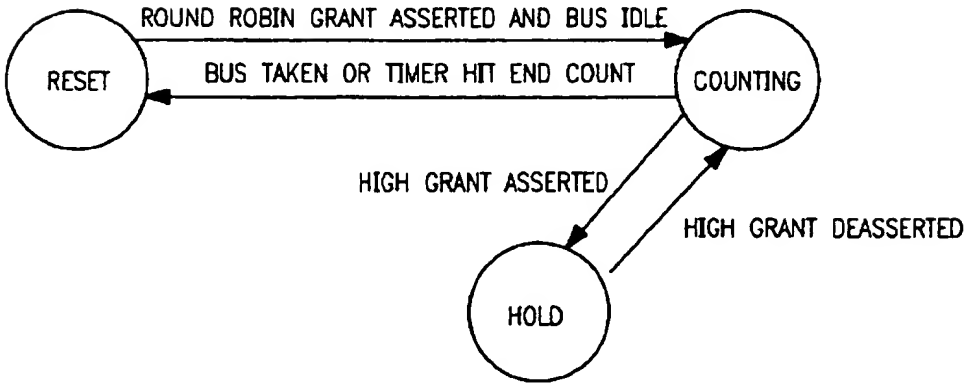

ROUND ROBIN GRANT ASSERTED AND BUS IDLE
BUS TAKEN OR TIMER HIT END COUNT
RESET
COUNTING
HIGH GRANT ASSERTED
HIGH GRANT DEASSERTED
HOLD

FIG. 4

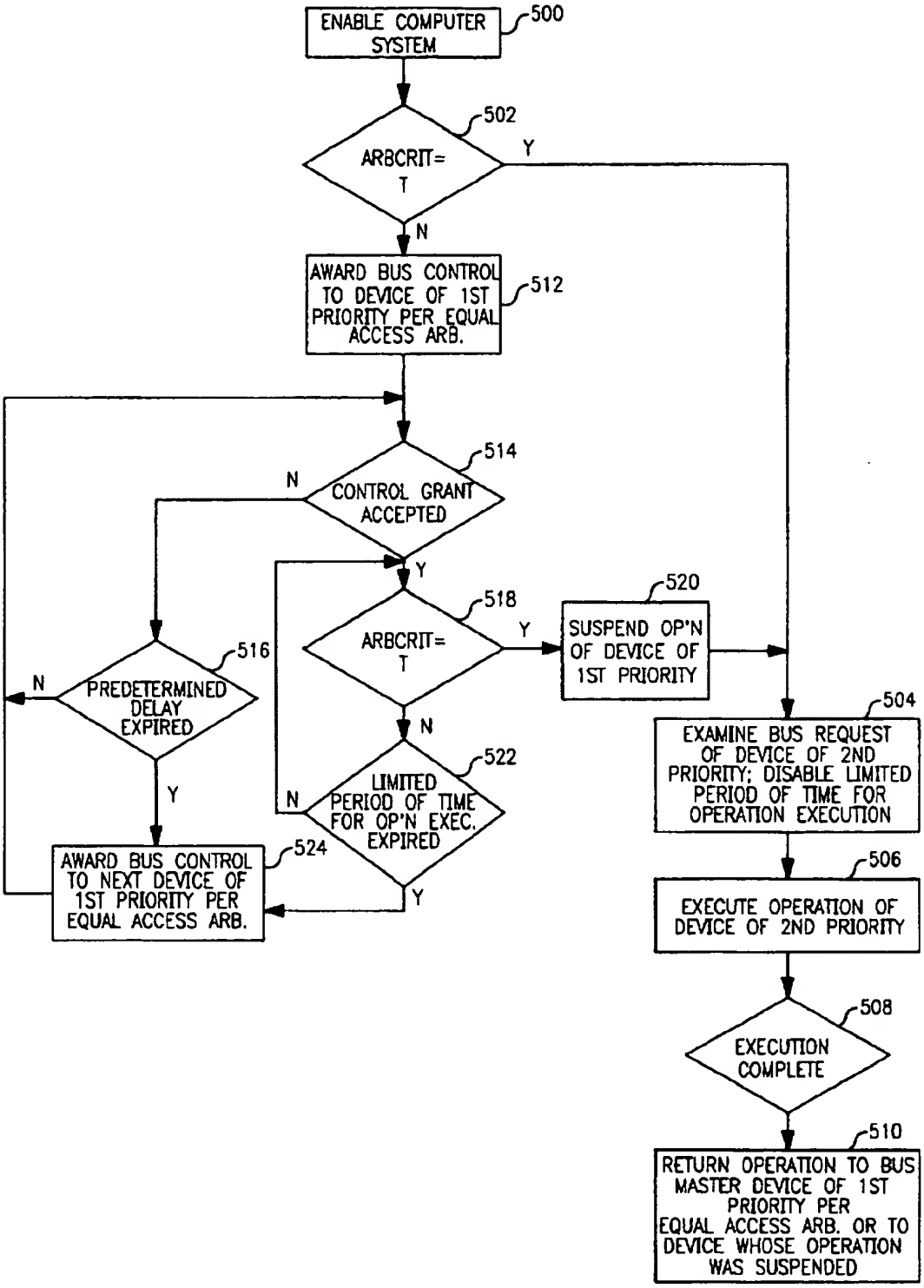

ENABLE COMPUTER SYSTEM
500
ARBCRIT= T
502
Y
N
AWARD BUS CONTROL TO DEVICE OF 1ST PRIORITY PER EQUAL ACCESS ARB.
512
CONTROL GRANT ACCEPTED
514
N
Y
ARBCRIT= T
518
Y
SUSPEND OP'N OF DEVICE OF 1ST PRIORITY
520
PREDETERMINED DELAY EXPIRED
516
N
Y
LIMITED PERIOD OF TIME FOR OP'N EXEC. EXPIRED
522
N
Y
AWARD BUS CONTROL TO NEXT DEVICE OF 1ST PRIORITY PER EQUAL ACCESS ARB.
524
EXAMINE BUS REQUEST OF DEVICE OF 2ND PRIORITY; DISABLE LIMITED PERIOD OF TIME FOR OPERATION EXECUTION
504
EXECUTE OPERATION OF DEVICE OF 2ND PRIORITY
506
EXECUTION COMPLETE
508
RETURN OPERATION TO BUS MASTER DEVICE OF 1ST PRIORITY PER EQUAL ACCESS ARB. OR TO DEVICE WHOSE OPERATION WAS SUSPENDED
510

FIG. 5

METHOD AND APPARATUS FOR ARBITRATING ACCESS TO A SHARED BUS

BACKGROUND OF THE INVENTION

1. Field Of The Invention

The present invention relates generally to computer system architecture and control, and more particularly to a method and apparatus for arbitrating the access of multiple devices in a computer system to a shared bus of the computer system.

2. State of the Art

As computer systems become increasingly more complex, the manner by which they access and control shared devices becomes a significant factor in maintaining high system throughput and operating efficiency. For example, as the number of devices that share a common bus in a computer system is increased, the computer system must more efficiently arbitrate control of the bus among the devices.

Known techniques for arbitrating access to a shared bus have been deemed acceptable, given the relatively small number of devices and the absence of time critical operations, such as real time signal processing, in conventional computer systems. However, these techniques have effectively limited the number and types of devices which can be connected to tie shared bus without affecting processing efficiency. For example, sophisticated multimedia features cannot be included in conventional computer systems without imposing a trade-off in overall operating efficiency of the computer system. This is because computer systems which provide multimedia features must be able to process large quantities of data in real time; for example, real time audio and video data streams.

Conventional techniques for arbitrating control of a shared bus in a computer system are of two general types: (1) those which cannot assign a high priority to a device that performs time critical operations, such as real time signal processing; and (2) those which can assign hierarchical priorities to all devices of the computer system but which, in so doing, allow the higher priority devices to dominate control of the computer system and effectively deny lower priority devices from obtaining control of the shared bus. Currently available arbitration techniques include: (1) first-in/first-out arbitration; (2) "daisy-chaining"; and (3) use of a central arbiter. With first-in/first-out arbitration, individual requests from the plurality of devices sharing a common bus in a computer system are queued in the order in which they arrive at an arbiter of the computer system. Such a scheme ensures equal access among the plurality of devices to the bus so that no one device will be denied access to the bus for an extended period. The disadvantage of first-in/first-out arbitration is that it does not take into account the increasingly more diverse and sophisticated architecture of computer systems, such as multimedia computer systems, wherein the range of features and overall operating efficiency can be enhanced by providing different devices of the computer system different priorities in accessing the shared bus.

Daisy chaining is similar to first-in/first-our arbitration in that a fixed routine is used to arbitrate requests from plural devices of a computer system so that no single device will be denied control of the bus over extended periods of time. With daisy chaining, a bus available signal indicating that the shared bus is available for use is transmitted from one device to the next in a predetermined order. Once a device which is requesting access to the bus receives the bus available signal, that device can access and/or take control

of the bus. As with first-in/first-out arbitration, daisy chaining can ensure that all devices will have an opportunity to acquire control of the shared bus. However, this advantage is acquired at the expense of overall operating efficiency when, for example, multimedia features are provided. For example, a relatively low level device which could process data at a later time can be awarded control of the bus even though a real time signal is being received which requires immediate attention.

Computer systems which use a central arbiter to arbitrate access to a shared bus among a plurality of devices typically afford each device requesting access to the bus a given hierarchical priority. This priority can be determined on the basis of predetermined criteria, such as; (1) the importance of the operation to be executed by the device, relative to those of other devices; and (2) the time which the device has been waiting to gain control of the shared bus. Using the priority of each device requesting control of the shared bus, the central arbiter will queue the various requests from the plural devices. This queuing of requests can, of course, be reconfigured each time a new request is received.

An advantage to the use of a central arbiter is that the plurality of devices included in a computer system can be attributed varying priorities in gaining access to and control of a shared bus, such that drawbacks associated with first-in/first-out arbitration and daisy chaining are avoided. However, prioritization techniques based on the use of a central arbiter can be relatively complex and can often result in one or more lower priority devices of the computer system being effectively denied access to the shared bus for extended periods of time. For example, the possibility that one or more devices would be denied control of the bus is quite high in multimedia based systems, wherein devices which execute real time operations can dominate system bus control; when these highest priority devices are idle, system bus control is awarded to the next lower priority devices. Only when all other devices are idle do the very lowest priority devices, acquire system bus control. Accordingly, the very lowest level devices would have great difficulty ever obtaining system bus control. Thus, while the use of a central arbiter overcomes the drawbacks of first-in/first-out arbitration and daisy chaining, it suffers the drawback of effectively denying bus system control to some devices.

Accordingly, it would be desirable to provide a computer system wherein the control of a shared bus by a plurality of devices included in the computer system is provided in a manner whereby overall operating efficiency is enhanced without effectively denying one or more devices in the computer system from control of the bus for extended periods of time. Further, it would be desirable to provide a computer system wherein such enhanced operating efficiency can be achieved even with an increased number of devices included in the computer system having access to the shared bus. Further, it would be desirable to provide enhanced operating efficiency in a computer system which provides sophisticated multimedia features, such as real time audio/video data processing, animation and so forth.

SUMMARY OF THE INVENTION

The present invention is directed to providing a computer system which arbitrates control of a shared bus among plural devices included in the computer system. In accordance with the present invention, at least one of the devices is afforded a higher priority than the remaining devices, yet none of the remaining devices are effectively denied system bus access or Control for extended periods of time. The present inven-

tion can therefore increase operating efficiency even as the number of devices included in the computer system is increased to achieve enhanced processing power. In addition, the present invention can provide sophisticated multimedia features, including real time signal processing, without sacrificing overall operating efficiency. In accordance with the present invention, the plural devices arbitrate system bus control in a manner which achieves acceptable multimedia results when processing real time data streams such as video data streams, audio data streams, animation data streams, and so forth, yet which does not sacrifice the access of remaining devices in the computer system to the shared bus.

In accordance with exemplary embodiments of the present invention, at least one device performs time-critical operations such as inputting/outputting serial or real time data streams, and can request relatively immediate bus system control via a dedicated signal line The remaining devices are lower-priority devices which can request the system bus control via an equal access arbitration scheme whereby all of the lower priority devices have an equal chance of acquiring system bus control, and via the use of standard bus request lines. In an absence of a signal on the dedicated signal line from the at least one device, system bus control is awarded to one of the lower level devices using the equal access arbitration scheme.

In accordance with exemplary embodiments, a computer system for processing data includes, a system bus for transferring signals within said computer system; a plurality of devices of a first priority, each of said devices of first priority generating a first signal to request control of said system bus; at least one device of a second priority for generating a second signal to request control of said system bus; and an arbiter responsive to said first signal from each of said plurality of devices of first priority, and to said second signal from said at least one device of second priority, for arbitrating control of said system bus among said plurality of devices of first priority using equal access arbitration in an absence of detecting said second signal and, upon detecting said second signal, granting control of said system bus to said at least one device of second priority.

Further, exemplary embodiments are directed to a method for processing data in a computer system which includes multiple devices connected with a common system bus, comprising the steps of, assigning a first plurality of devices included in said computer system a first priority, each of said plurality of devices generating a first signal to request control of said system bus; assigning at least one additional device included in said computer system a second priority, said at least one additional device generating a second signal to request control of said system bus; and controlling access to said system bus among said first plurality of devices and said at least one device, said step of controlling, further including the steps of: arbitrating control of said system bus among said plurality of devices of first priority using equal access arbitration in an absence of said second signal; and granting control of said system bus to said at least one device of second priority upon detecting said second signal.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features and advantages of the invention will be readily apparent to those skilled in the art from the following written description, when read in conjunction with the drawings, in which:

FIG. 1 illustrates an exemplary embodiment of a computer system with a plurality of devices attached to a high performance bus,

FIG. 2 illustrates a computer system according to an exemplary embodiment of the present invention;

FIG. 3 illustrates an equal access arbitration scheme according to an exemplary embodiment of the present invention;

FIG. 4 illustrates an arbitration method according to an exemplary embodiment of the present invention; and

FIG. 5 illustrates an exemplary flowchart relating to an arbitration method in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 illustrates a computer system for processing, data, the computer system having a plurality of devices which communicate over a shared system bus 12. In the exemplary FIG. 1 embodiment, the system bus which is used to transfer signals can be, for example, a Peripheral Component Interconnect (PCI) local bus. A PCI Local Bus is described in "PCI Local Bus Specification," Review Draft Revision 2.1, published Oct. 21, 1994, by PCI Special Interest Group, the disclosure of which is hereby incorporated by reference in its entirety. However, it will be appreciated by those skilled in the art that the present invention is not limited to a PCI local bus, but can be used with any high performance bus for interconnecting plural devices, such as highly integrated peripheral control devices, peripheral add-on boards, processor/memory devices and so forth.

As illustrated in FIG. 1, a plurality of devices are illustrated which can request control of the system bus 12 to read or write data from or to another device. For purposes of the following discussion, any device which can request access to or control of the system bus 12 will be referred to as a bus master device. Those skilled in the art will appreciate that any other devices which do not request system bus control can also be connected to the system bus 12. As illustrated in FIG. 1, the plurality of bus master devices can include, but is not limited to, a processor 14 and associated processor bridge 16, a real time input/output (I/O) device 24, and a non-real real time I/O device 26.

The processor bridge 16 allows a processor 14, such as the main processor of a computer system, to access the other devices connected to the PCI bus. One example of this type of access is when the processor 14 performs read or write operations to registers that are contained in either the real time I/O device 24 or the non-real time I/O device 26. The processor bridge 16 also allows either of the I/O devices 24 and 26 to access a system memory 20 via a memory controller 18 in known manner.

According to exemplary embodiments of the present invention, each of a plurality of the devices is assigned a first priority. At least one additional device is assigned a second priority. According one embodiment of the present invention, the bus master device assigned the second priority is the bus master device which handles user specified priority operations, such as time critical processing of serial or real time data streams, including video data streams, audio data streams, animation data streams, and so forth. The operations assigned second priority operations are typically operations which need special attention by the computer system, such as real time transfers of data which should not be interrupted for a given period of time.

For example, if a video data stream is being transferred from an external device through the computer system to a display screen, interruptions in the transfer of the video data stream to the computer system will be noticeable to an observer of the display screen. In the exemplary embodi-

ment of FIG. 1. the real time I/O device 24 is therefore designated as the bus master device of second priority. In the exemplary FIG. 2 embodiment. the real time input/output device 24 is represented as a video input direct memory access (DMA) controller 31 which serves as the bus master device of second priority. The other bus master devices shown. namely the processor bridge 16 and the non-real time I/O device 26. are assigned the first lower priority since they are not performing time critical operations; rather. the devices of the first lower priority can. for example. perform any non-real time operation.

The protocol for gaining control of the system bus is termed arbitration and is overseen by an arbiter means. represented as an arbiter 22. for arbitrating control of the system bus among the plurality of devices of first priority and the at least one device of second priority. It will be appreciated by those skilled in the art. that the arbiter 22 can be located anywhere throughout the computer system. The arbiter 22 has a plurality of request lines 28 coming in from the bus master devices and a plurality of grant lines 30 which are each connected to one of the bus master devices. The arbiter can. in accordance with an exemplary embodiment. include a processor 23 for performing the various arbitration functions based upon one or more user configurable arbitration schemes. Those skilled in the art will appreciate that conventional programming techniques can be used to program the arbiter to implement the functionality described herein.

According to the exemplary embodiment illustrated in FIG. 2. the arbiter 22 is responsive to a first signal from each of the plurality of devices of first priority and to a second signal from the at least one device of second priority to arbitrate control of the system bus. The arbiter 22 is. in an exemplary embodiment. connected to each of the bus master devices by the request line 28 and by a grant line 30. Those skilled in the art will appreciate that a single bi-directional line can alternately be used. if desired. for the request and grant signals. According an exemplary embodiment of the present invention. the bus master devices of first priority can request control of the system bus by asserting their request lines 28. If the arbiter grants the request. the arbiter sends a grant signal to the requesting bus master device. In order to ensure that all of the bus master devices of first lower priority have an equal opportunity to use the bus. the arbiter uses an equal access arbitration scheme. For example. a round robin arbitration scheme which awards system bus control to each of any user-specified bus master devices of first priority in a fixed. sequential order. can be used as an equal access arbitration scheme. Alternately. any arbitration scheme which ensures equal access among the bus master devices. such as token ring protocol. can be used as well.

Each bus master device of first priority is only granted a user-configurable limited period of time to control the bus with each request signal. As a result. if a given bus master device of first priority can not complete the number of operations associated with a given transaction within the limited period of time it has been assigned the bus. the given bus master device will have to make another request which will eventually be granted by the arbiter 22 pursuant to the round-robin arbitration scheme. The limited period of time can be configured by the user to take criteria such as the number of devices connected to the system bus into account.

A possible arbitration scheme is illustrated in FIG. 3 which has five bus master devices labelled A–E of first priority. As illustrated in FIG. 3 by arrows 1–5. if all five of the bus master devices of first priority are requesting access to the bus. the arbiter first grants access Lo bus master device A followed in sequence by bus master devices B.C.D. and B. If all of the non-priority bus master devices keep requesting the bus. the arbiter will continue to grant access in this rotating. fixed sequence. fashion. However. should a given one of the bus master devices A–E not request access to or control of the bus at the time the arbiter would otherwise offer control to that given bus master device. the arbiter simply skips that bus master device and offers system bus control to the next sequential bus master device which has asserted its bus request signal.

For example. if only bus master devices A and B are requesting access. the arbiter will rotate access to e bus just between the non-priority bus master devices A and B as illustrated by arrows I and 6. Furthermore. if bus master devices A. C. and E are requesting access to the bus. the arbiter will rotate access to the bus between bus master devices A. C. and E as illustrated by arrows 11. 13. and 5. Thus. a fixed rotational sequence is used to sequentially grant system bus control among only those bus master devices of first priority which are asserting their bus request signal.

In addition to the request line 28 and the grant line 30. the bus master device of second priority also has an arbitration critical line 32 (labelled ARBcrit in FIG. 2) connected to the arbiter 22. In an exemplary embodiment. when the bus master device of second priority wants control of or access to the system bus. the bus master device of second priority asserts its request line as well as the ARBcrit line. Upon receiving the request and ARBcrit signals. the arbiter then grants control of the bus to the bus master device of second priority without regard to the equal access arbitration scheme.

According to an exemplary embodiment. the bus master device of second priority can keep control of the bus for as long as the bus master device of second priority asserts the ARBcrit signal. In other words. the ARBcrit signal blocks. or inhibits. the arbiter 22 from terminating control of the bus by the bus master device of second priority at the end of the user-configurable limited period of time normally granted to a requesting bus master device. As a result. the bus master device of second priority can perform all of the necessary operations for completing transfer of serial or real time data streams without losing control of the bus.

According to an exemplary embodiment of the present invention. the arbiter 22 can immediately grant control of the bus to the bus master device of second priority. In alternate embodiment. the arbiter 22 can delay the grant for a predetermined period of time. for example. at least a sufficient amount of time for the bus master device currently in control of the bus to complete the single instruction it is currently executing and buffer status information. For example. the delay can be set to three or more clock cycles in an exemplary embodiment. This delay gives a bus master device of first priority presently using the bus an opportunity to complete a current operation before the bus master device of second priority is granted control of the bus. and thereby avoids the bus master device of first priority from entering a deadlock situation.

According to an exemplary embodiment of the present invention as described above. bus masters of first priority can be guaranteed an opportunity to complete one transaction (for example. a transaction including one or more instructions) when granted control of the bus for the predetermined limited period of time. assuming that the ARBcrit signal is not asserted. This ensures that the remaining bus master devices of first priority are offered a fair chance to

use the bus. As illustrated in FIG. 4, if a bus master device of first priority receives a grant from the arbiter to take control of the system bus, the bus master device granted control has a predetermined period of time during which it can take control of the bus. Again, this predetermined period of time is user-configurable. In the exemplary FIG. 2 embodiment, this predetermined period can be a value on the order of 16 clock cycles of the idle time as defined by the PCI Local Bus specification and the current count can be tracked with a counter.

More particularly, idle time can be defined as the IRdy and Frame signals being deasserted (high) in accordance with PCI bus protocol. If the bus master device of second priority requests control of the bus during this period of time, the bus master device of second priority is granted access to the bus. In this case, the arbiter can, if desired, be configured to hold, or store, a count representing the elapsed idle time of a bus master device previously granted control of the system bus until the bus master device of second priority releases control of the bus. Then, the incrementing of the count resumes where it left off and continues to the predetermined value. If the count reaches the predetermined value, an internal state bit in the arbiter can be set to indicate that the bus master device of first priority is attempting to hold the bus. This bit allows the arbiter to advance to the next requesting bus master device of first priority without allowing a transaction from bus master device previously granted system bus control.

FIG. 5 illustrates an exemplary flowchart of an operation associated with the exemplary embodiment illustrated in FIG. 2. In FIG. 5, the computer system is enabled at block **500**. The ARBcrit signal is examined in decision block **502** to determine whether the bus master device of second priority has requested system bus control. If so, operation proceeds to block **504** wherein the bus request line of the bus master device of second priority can be examined. Further, the predetermined limited period of time for executing an operation can be disabled so that the bus master of second priority can retain operation of the system bus for as long as the ARBcrit signal is asserted.

In block **506**, the bus master device of second priority executes operations, such as time-critical operations. Decision block **508** represents an examination of the ARBcrit signal to determine whether the bus master device of second priority has completed executing its time-critical operations. If not, the bus master device of second priority is permitted to complete such operations. However, if such operations are complete, flow returns via block **510** to the bus master device of first priority which had previously been in control of the system bus at the time the ARBcrit signal was asserted. Of course if the ARBcrit signal was asserted at power up (i.e., block **502**), system bus control is supplied to the first bus master of first priority which is to receive control pursuant to the equal access arbitration scheme.

Returning to the decision block **502**, if the ARBcrit signal was not asserted at the time the system was enabled, then the arbiter will award priority to the first bus master of first priority in accordance with the equal access arbitration scheme in block **512**. In block **514**, it is examined whether the bus master selected by the arbitration scheme has accepted the grant of the system bus. If not, decision block **516** examines whether the predetermined time delay has expired. If so, operation flows to block **524** to select the next bus master of first priority pursuant to the equal access arbitration scheme.

Returning to decision block **514**, assuming that the bus master device of first priority selected by the arbiter has been granted control of the system bus, and that the bus master device has accepted such control, the ARBcrit signal is monitored in decision block **518**. Again, if the ARBcrit signal becomes true at any time while another bus master device of first priority is in control of the bus, operation proceeds to block **520** wherein operation of the bus master device of first priority is suspended, and operation flows to blocks **504–510**.

Assuming that the ARBcrit signal has not been asserted in block **518**, operation flows to decision block **522** wherein the predetermined, limited period of time during which a bus master device of first priority can execute control of the system bus is monitored. Provided this period has not expired, monitoring of the ARBcrit signal and the predetermined, limited period of time continue. However, once the predetermined, limited period of time has expired, operation of the bus master device of first priority currently in control of the system bus is terminated, and operation returns to block **524**, wherein the next bus master device of first priority is selected pursuant to the equal access arbitration scheme.

Those skilled in the art will appreciate that the present invention is not limited to the exemplary embodiments described above. For example, those skilled in the art will appreciate that the bus master device of second priority need not include a separate request line and a separate arbitration critical line, but that an arbitration critical line alone can be used. In this case, the ARBcrit signal can both serve as a request signal and as an inhibit signal for inhibiting the limited period of time afforded the bus master devices of first priority to retain system bus control in response to a single bus request signal.

Further those skilled in the art will appreciate that the computer system of FIG. 3 can be configured such that the bus master device of second priority executes both time-critical (for example, real dime) signal processing operations and non-real time operations. In the case of non-real time operations, the bus master device of second priority can request access to the system bus using its request line, and can be awarded system bus control via the equal access arbitration scheme. However, in the case of real time operations, system bus control can be acquired via use of the arbitration critical line.

In addition, those skilled in the art will appreciate that while each bus request line is illustrated as a hardwired input to an input port of the arbiter, these inputs can be configured and reconfigured by the user in any desired order. Alternately, a reconfiguring of order can be achieved using software reassignment of the imput ports to the arbiter.

Further, those skilled in the art will appreciate that the limited period of time, as well as any predetermined time periods or values described above, can be configured and reconfigured by the user at any time. For example, as additional devices are added to the computer system the limited period of time can be reduced.

Those skilled in the art will also appreciate that although the exemplary FIG. 1 embodiment has been described as including a single real time input/output device **24**, any number of such devices can be included in the computer system. Where more than one real time device is included in the computer system, the request lines from all such devices can be used to assert the ARBcrit signal. That is, when any of the real time devices asserts a request to control the bus, the ARBcrit signal is asserted. To accommodate a situation where more than one real time device has asserted a request for bus control, an arbitration scheme, separate from the

equal access arbitration scheme associated with non-real time devices, can be used to award system bus control to one of the real time devices.

For example, each of the real time devices can be afforded a hierarchical priority as a function of the bandwidth associated with signals processed by that device. More particularly, a device associated with a high bandwidth video signal can be afforded highest priority. A device associated with a mid-bandwidth audio signal can be afforded a lower priority. Another device used to process real time serial data streams can be afforded a next lower priority.

The priority of a real time device requesting system bus control can be identified by an input/output device which is either separate from or included within the arbiter. As a result, upon requesting system bus control, the ARBcrit line will be serviced by the arbiter to award system bus control to the highest priority real time device currently requesting bus control.

If more than one real time device is currently requesting system bus control, then the highest priority device is awarded system bus control first. After the operations of that device are complete, the ARBcrit signal remains asserted, and the next lower priority real time device is awarded system bus control. This process is repeated until all real time devices have completed their operations, and the ARBcrit signal has been deasserted.

Those skilled in the art will appreciate that in place of a hierarchical priority scheme, any arbitration scheme can be used to award priority to one of the real time devices. Further, those skilled in the art will appreciate that multiple arbitration schemes can be used to service the real time devices. For example, multiple devices for processing video signals, can be collectively designated the highest priority devices, and then awarded system bus control in a round robin format. After all such devices have finished their operations, system bus control can be awarded to the next lower priority device or devices.

It will be appreciated by those of ordinary skill in the art that the present invention can be embodied in other specific forms without departing from the spirit or essential character thereof. The presently disclosed embodiments are therefore considered in all respects to be illustrative and not restrictive. The scope of the invention is indicated by the appended claims rather than the foregoing description, and all changes which come within the meaning and range of equivalents thereof are intended to be embraced therein.

What is claimed is:

1. A computer system for processing data comprising:

a system bus for transferring signals within said computer system;

a plurality of devices of a first priority, each of said devices of first priority generating a first signal to request control of said system bus;

at least one video device of a second priority for generating a second signal to request control of said system bus; and

an arbiter responsive to said first signal from each of said plurality of devices of first priority, and to said second signal from said at least one video device of second priority, for arbitrating control of said system bus among said plurality of devices of first priority using an equal access arbitration in an absence of detecting said second signal and, upon detecting said second signal, granting control of said system bus to said at least one video device of second priority; wherein

an elapsed period of time subsequent to a grant of system bus control to one of said plurality of devices of first priority is stored while the at least one video device of second priority uses the system bus.

2. A computer system according to claim 1, wherein said plurality of devices of first priority are granted access to the system bus in a round-robin format.

3. A computer system according to claim 1, wherein said at least one video device of a second priority is granted control of the system bus for as long as the second signal remains asserted.

4. A computer system according to claim 1, wherein said plurality of devices of first priority have a predetermined period of time in which to acquire control of said system bus following a grant of system bus control by said arbiter.

5. A computer system according to claim 2, wherein said plurality of devices of first priority are granted control of the system bus for a limited period of time in response to each request for system bus control.

6. A computer system according to claim 1, wherein said elapsed period of time is further incremented after said one of the plurality of devices of first priority, is again granted system bus control subsequent to said at least one video device of second priority having released system bus control.

7. A computer system according to claim 1, wherein the arbiter waits a predetermined period of time before granting bus access to said video device of second priority after a request by said video device of second priority.

8. A computer system according to claim 1, further comprising:

a plurality of devices of said second priority, said second signal being generated in response to a request from any of said plurality of devices of second priority to control said system bus.

9. The computer system of claim 1, wherein the at least one video device of second priority generates the second signal to request control of the system bus with the second priority, and generates the first signal to request control of the system bus with the first priority, and the elapsed period of time is stored while the at least one video device of second priority uses the system bus with the second priority.

10. A computer system according to claim 7, wherein the arbiter waits a predetermined number of clock cycles before granting bus access to said video device of second priority.

11. A computer according to claim 8, wherein said arbiter awards control of said system bus to one of said plurality of devices of second priority using hierarchical arbitration.

12. Method for processing data in a computer system which includes multiple devices connected with a common system bus, comprising the steps of:

assigning a first plurality of devices included in said computer system a first priority, each of said plurality of devices generating a first signal to request control of said system bus;

assigning at least one additional video device included in said computer system a second priority, said at least one additional video device generating a second signal to request control of said system bus; and

controlling access to said system bus among said first plurality of devices and said at least one video device, said step of controlling further including the steps of:

arbitrating control of said system bus among said plurality of devices of first priority using equal access arbitration in an absence of said second signal;

granting control of said system bus to said at least one video device of second priority upon detecting said second signal; and

storing an elapsed period of time subsequent to said grant of system bus control to one of said first plurality of devices of first priority while the at least one additional video device of second priority uses the system bus.

13. Method according to claim 12, wherein said at least one additional video device of second priority performs time-critical operations.

14. Method according to claim 12, wherein said devices of first priority have a predetermined period of time in which to act upon a system bus grant.

15. Method according to claim 12, wherein said devices of first priority are granted control of the system bus for a limited period of time in response to each request for system bus control.

16. Method according to claim 12, further including a step of:

continuing to increment said elapsed period of time after said one of the plurality of devices of first priority is again granted system bus control subsequent to said at least one additional video device of second priority having released system bus control.

17. Method according to claim 12, further including a step of:

waiting a predetermined period of time before granting bus access to said at least one additional video device of second priority after a request by said at least one additional video device of second priority.

18. Method according to claim 12, further comprising a step of:

assigning a plurality of additional devices included in said computer system said second priority, said second signal being generated in response to a request for system bus control by any of said plurality of additional devices of second priority.

19. The method of claim 12, wherein the at least one video device of second priority generates the second signal to request control of the system bus with the second priority, and generates the first signal to request control of the system bus with the first priority, and the elapsed period of time is stored while the at least one video device of second priority uses the system bus with the second priority.

20. Method according to claim 18, further including a step of:

arbitrating control of said system bus among said plurality of devices of second priority using hierarchical arbitration.

* * * * *

US006151651A

United States Patent [19]

Hewitt et al.

[11] Patent Number: 6,151,651

[45] Date of Patent: Nov. 21, 2000

[54] **COMMUNICATION LINK WITH ISOCHRONOUS AND ASYNCHRONOUS PRIORITY MODES COUPLING BRIDGE CIRCUITS IN A COMPUTER SYSTEM**

[75] Inventors: **Larry Hewitt**; **Dale E. Gulick**, both of Austin, Tex.

[73] Assignee: **Advanced Micro Devices, Inc.**, Sunnyvale, Calif.

[21] Appl. No.: **09/098,360**

[22] Filed: **Jun. 17, 1998**

[51] **Int. Cl.**[7] **G06F 13/00**

[52] **U.S. Cl.** **710/129**

[58] **Field of Search** 710/128, 105, 710/129

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,422,883	6/1995	Hauris et al.	370/62
5,450,411	9/1995	Heil	370/352
5,621,898	4/1997	Wooten	710/117
5,640,392	6/1997	Hayashi	370/395
5,742,847	4/1998	Knoll et al.	395/866
5,758,105	5/1998	Kelley et al.	395/293
5,761,430	6/1998	Gross et al.	395/200.55
5,761,448	6/1998	Adamson et al.	710/104
5,873,998	2/1999	Chee .	
5,948,080	9/1999	Baker	710/37

OTHER PUBLICATIONS

Intel Corporation, "Accelerated Graphics Port Interface Specification", Revision 1.0, Jul. 31, 1996, pp. 1–152.

Wickelgren, Ingred J., "The Facts About Fire Wire", Apr. 1997, pp. 20–25.

Glaskowsky, Peter N., "Cyrix Creates Ultimate CPU for Games", Microdesign Resoures, Dec. 8, 1997, pp. 16–18.

Gwennap, Linley, "MediaGX Targets Low-Cost PCs", Microprocessor Report, vol. 11, No. 3, Mar. 10, 1997, pp. 1–6.

Compaq, Digital Equipment Corp., IBM PC Company, Intel, Microsoft, NEC, and Northern Telcom, "Universal Serial Bus Specification", Revision 1.0, Jan. 15, 1996, pp. 3–268, particularly Chapters 4 and 5.

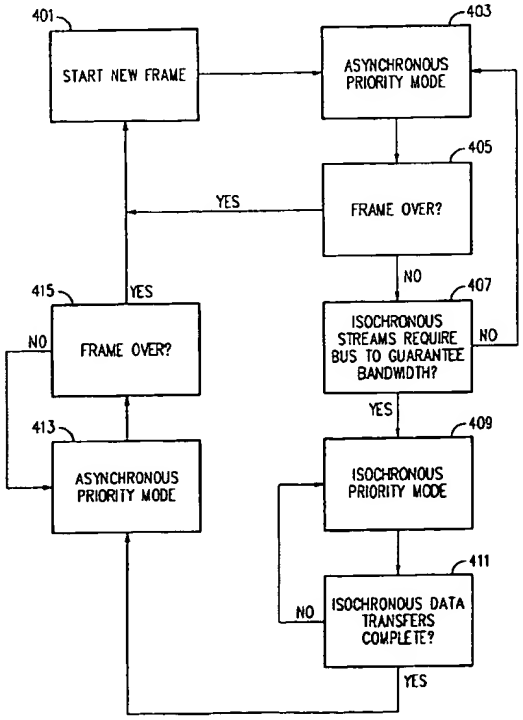
Primary Examiner—Glenn A. Auve

Attorney, Agent, or Firm—Zagorin, O'Brien & Graham, LLP

[57] **ABSTRACT**

A computer system includes a first processor integrated circuit. A first bridge integrated circuit is coupled to the processor via a host bus. The computer system includes an interconnection bus that couples the first bridge circuit to a second bridge circuit. The interconnection bus provides a first transfer mode for asynchronous data and a second transfer mode for isochronous data. The interconnection bus provides for a maximum latency and a guaranteed throughput for asynchronous and isochronous data.

24 Claims, 14 Drawing Sheets



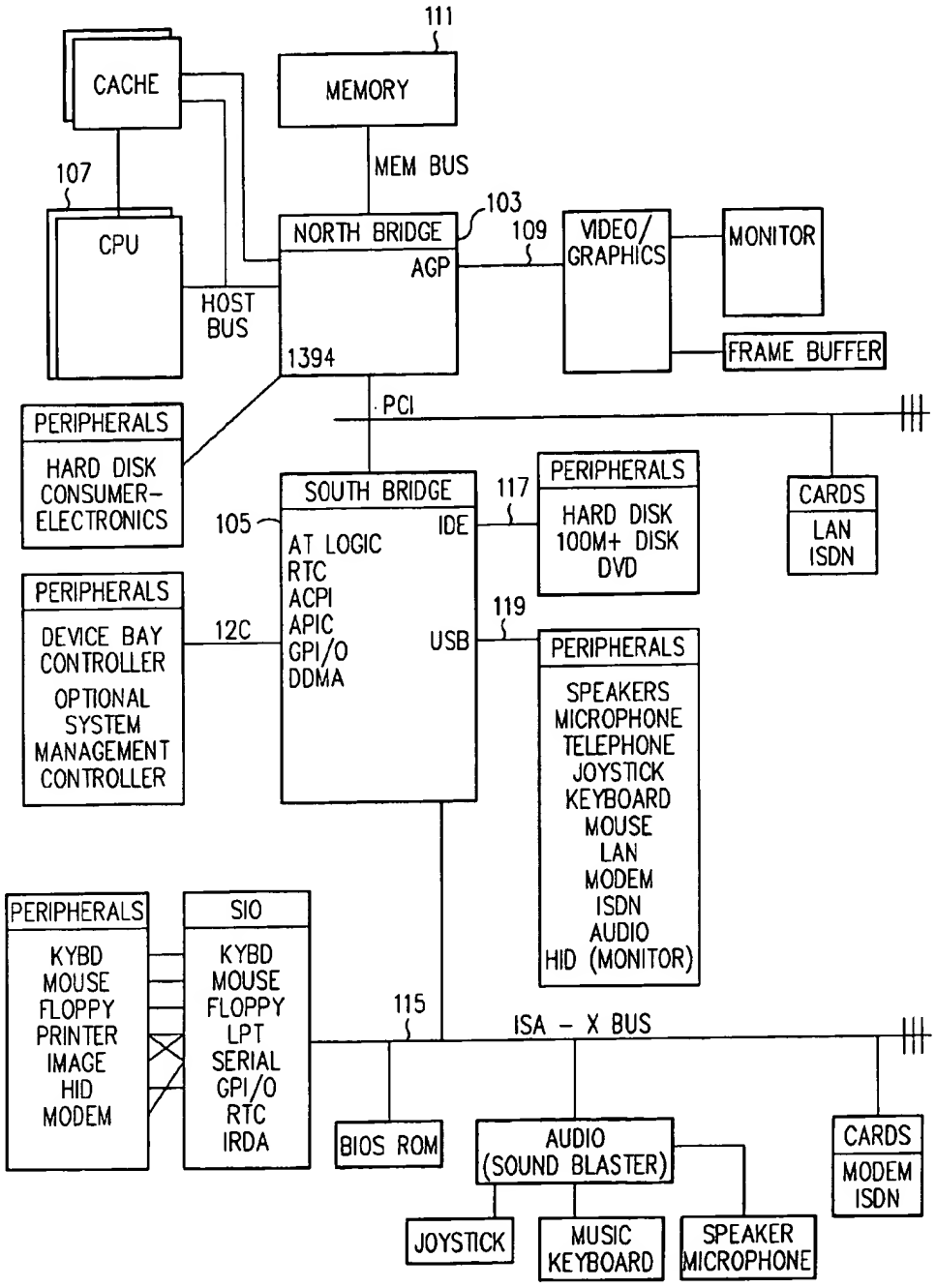


FIG. 1 (PRIOR ART)

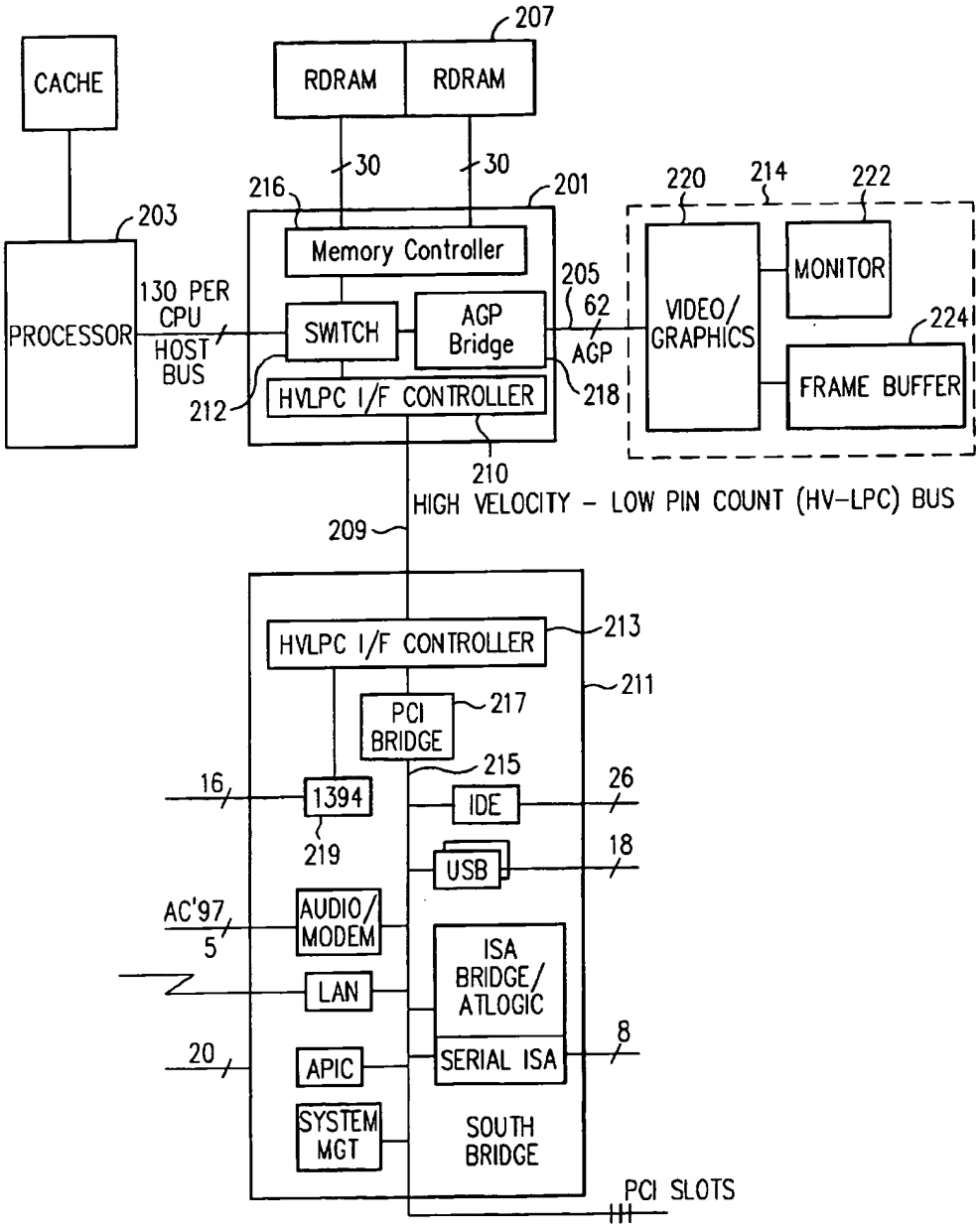

207
CACHE
RDRAM
RDRAM
30
30
216
201
220
214
222
203
Memory Controller
205
62
MONITOR
130 PER CPU
PROCESSOR
HOST BUS
SWITCH
AGP Bridge
AGP
VIDEO/ GRAPHICS
224
212
HVLPC I/F CONTROLLER
218
FRAME BUFFER
210
HIGH VELOCITY - LOW PIN COUNT (HV-LPC) BUS
209
HVLPC I/F CONTROLLER
213
211
PCI BRIDGE
217
215
16
1394
IDE
26
219
USB
18
AC'97
5
AUDIO/ MODEM
ISA BRIDGE/ ATLOGIC
LAN
20
APIC
SERIAL ISA
8
SYSTEM MGT
SOUTH BRIDGE
PCI SLOTS

FIG. 2

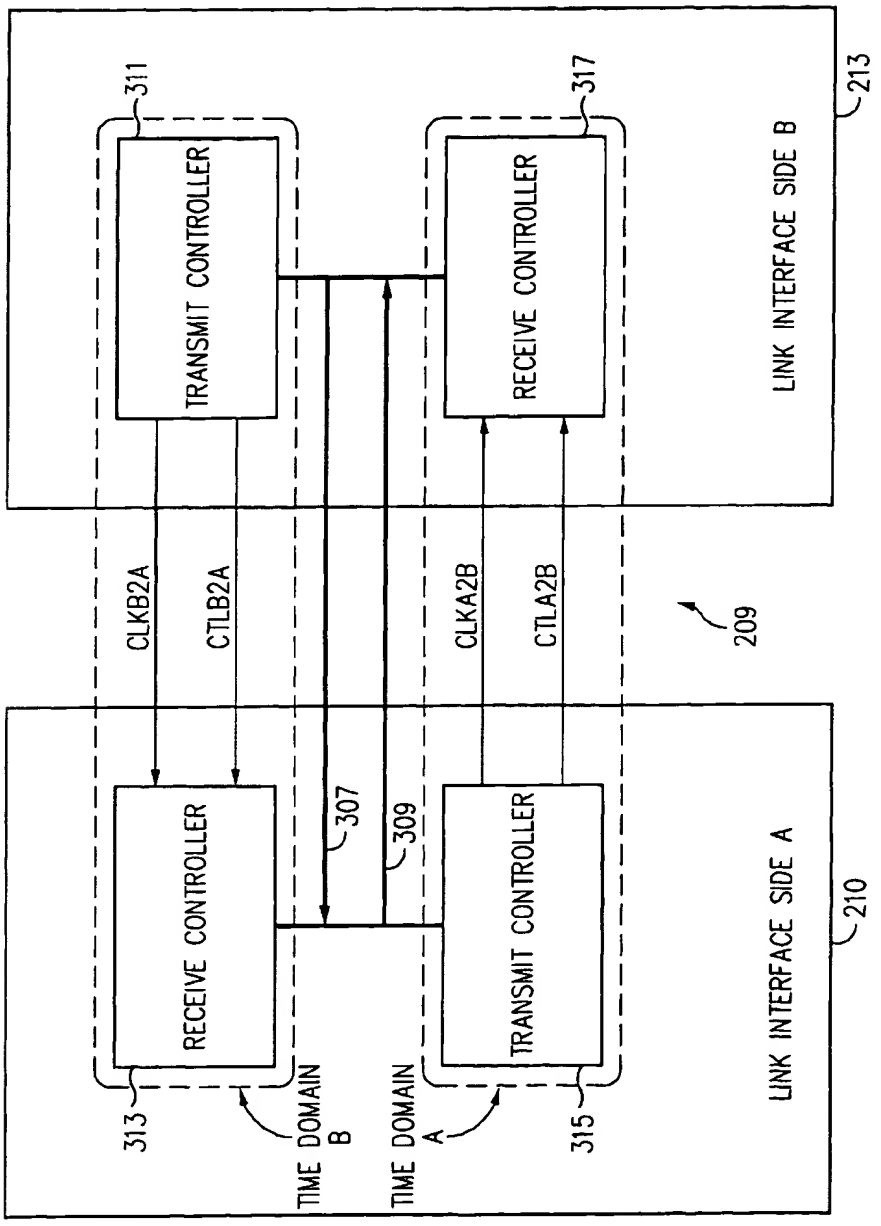

311
TRANSMIT CONTROLLER
317
RECEIVE CONTROLLER
213
LINK INTERFACE SIDE B
CLKB2A
CTLB2A
CLKA2B
CTLA2B
209
307
309
RECEIVE CONTROLLER
TRANSMIT CONTROLLER
313
315
TIME DOMAIN B
TIME DOMAIN A
LINK INTERFACE SIDE A
210

FIG. 3

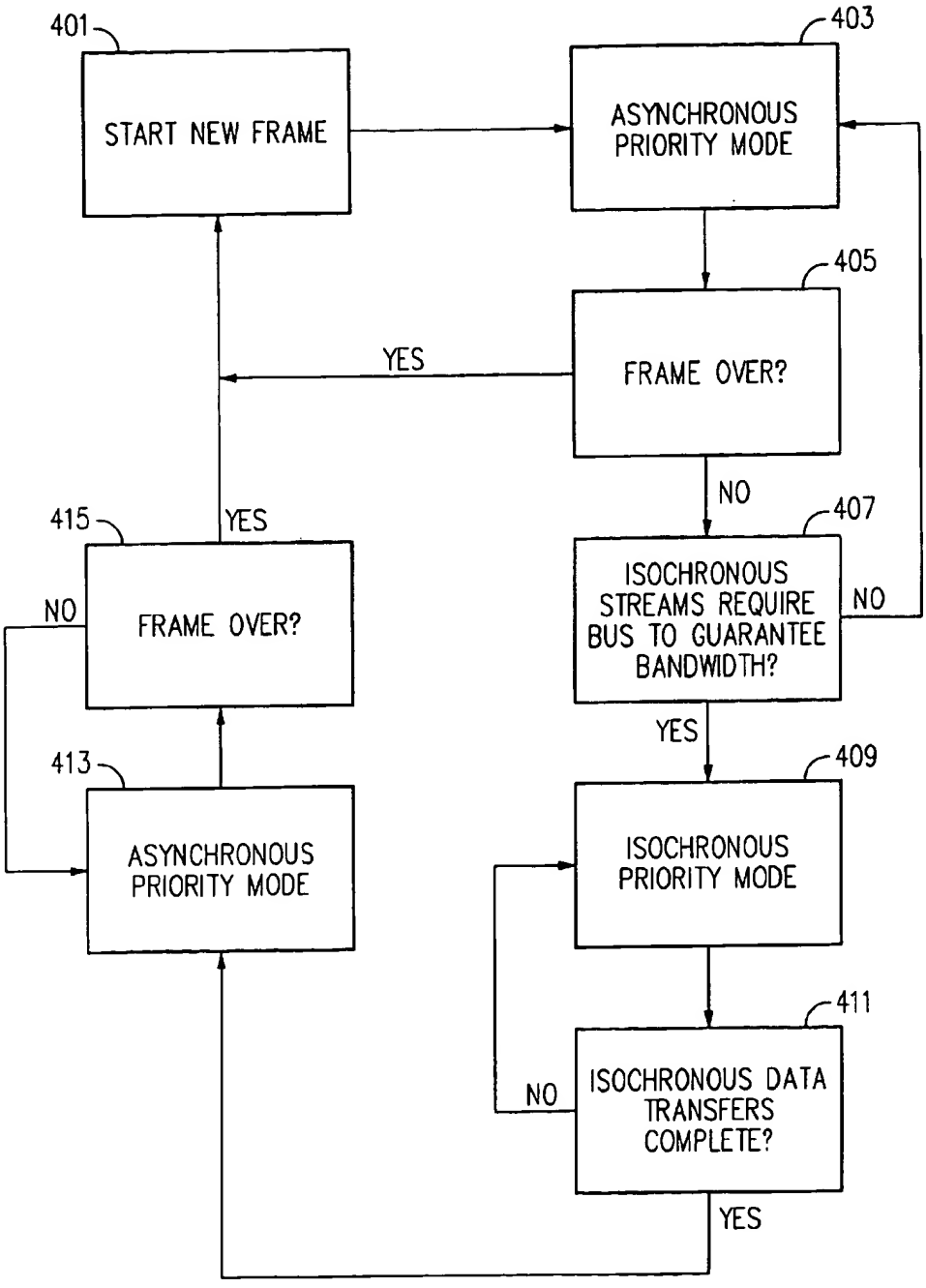

401
START NEW FRAME
403
ASYNCHRONOUS PRIORITY MODE
405
FRAME OVER?
YES
NO
407
ISOCHRONOUS STREAMS REQUIRE BUS TO GUARANTEE BANDWIDTH?
NO
YES
409
ISOCHRONOUS PRIORITY MODE
411
ISOCHRONOUS DATA TRANSFERS COMPLETE?
NO
YES
415
FRAME OVER?
YES
NO
413
ASYNCHRONOUS PRIORITY MODE

FIG. 4

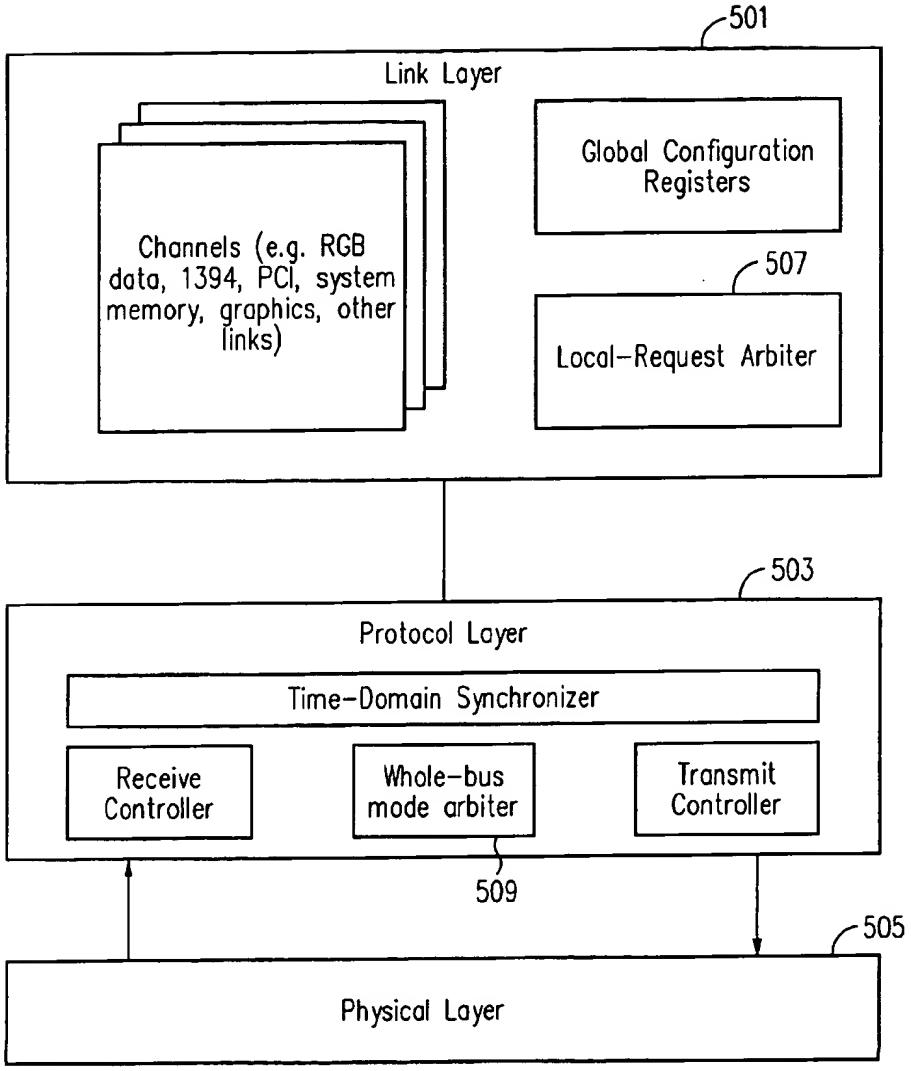

501
Link Layer
Channels (e.g. RGB data, 1394, PCI, system memory, graphics, other links)
Global Configuration Registers
507
Local-Request Arbiter
503
Protocol Layer
Time-Domain Synchronizer
Receive Controller
Whole-bus mode arbiter
Transmit Controller
509
505
Physical Layer

FIG. 5

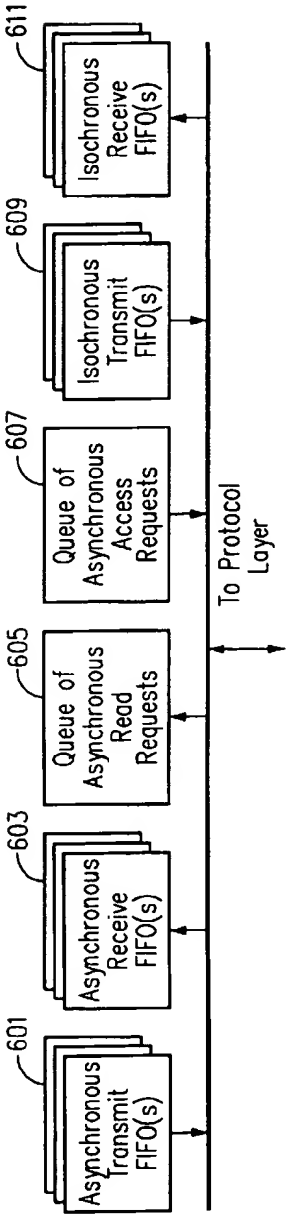

601
Asynchronous Transmit FIFO(s)
603
Asynchronous Receive FIFO(s)
605
Queue of Asynchronous Read Requests
607
Queue of Asynchronous Access Requests
609
Isochronous Transmit FIFO(s)
611
Isochronous Receive FIFO(s)
To Protocol Layer

FIG. 6

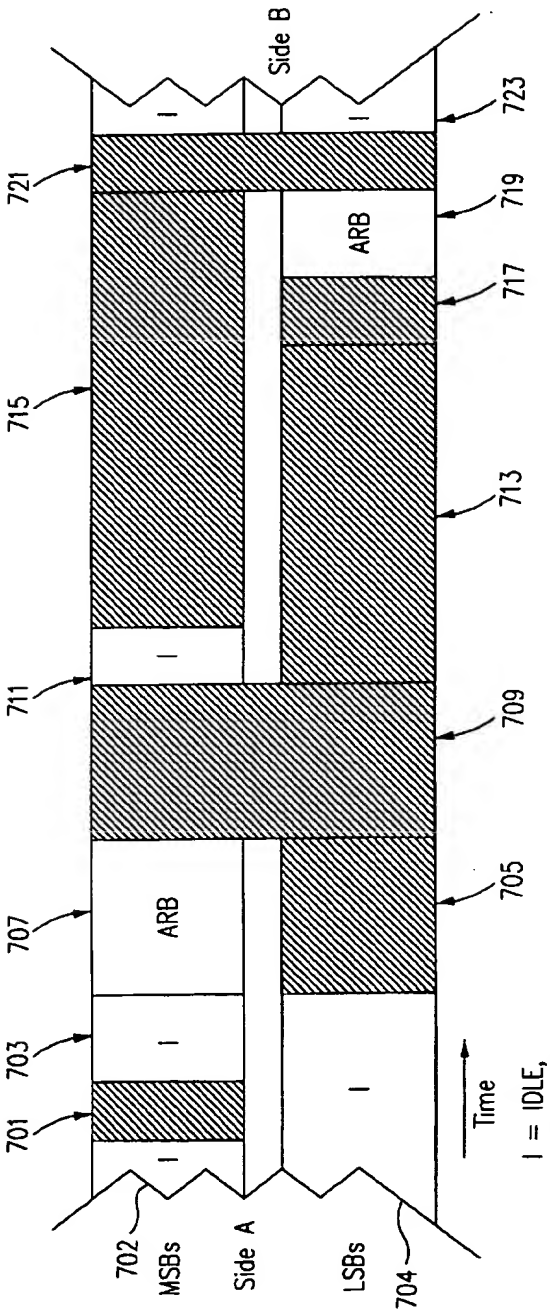

Side B
Side A
MSBs
LSBs
ARB
ARB
I
Time
I = IDLE,
ARB = ARBITRATE
= TRANSMIT DATA
701
702
703
704
705
707
709
711
713
715
717
719
721
723

FIG. 7

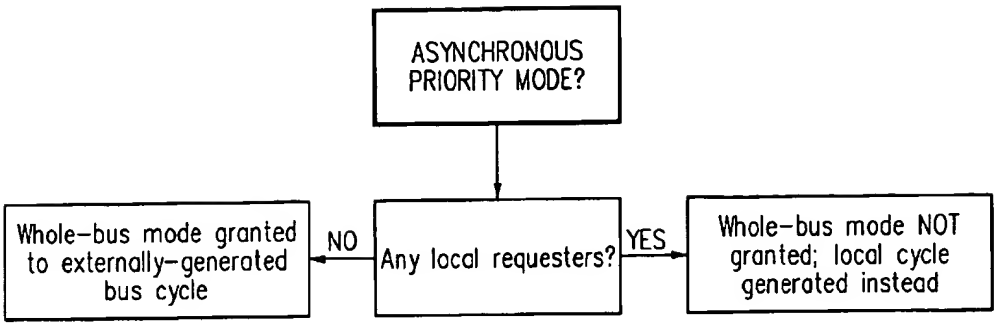

ASYNCHRONOUS PRIORITY MODE?
Any local requesters?
NO
YES
Whole-bus mode granted to externally-generated bus cycle
Whole-bus mode NOT granted; local cycle generated instead

FIG. 8

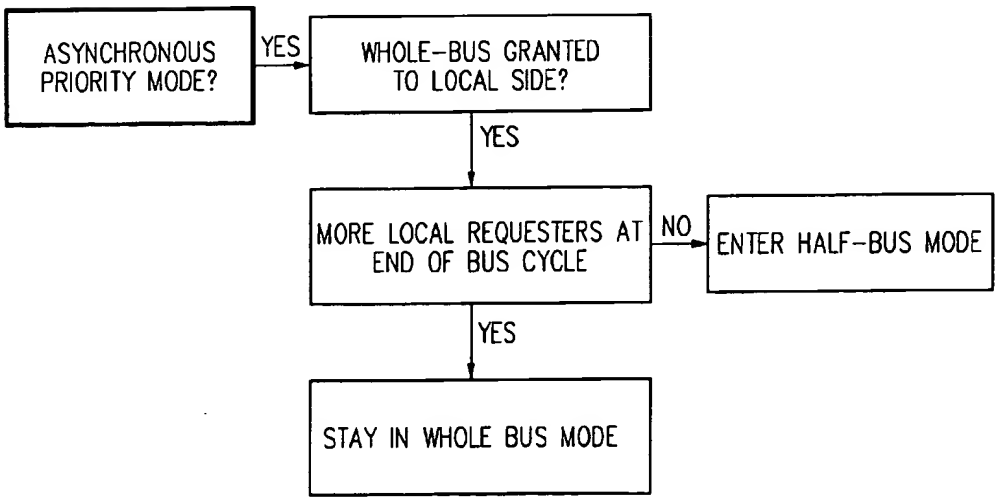

ASYNCHRONOUS PRIORITY MODE?
YES
WHOLE-BUS GRANTED TO LOCAL SIDE?
YES
MORE LOCAL REQUESTERS AT END OF BUS CYCLE
NO
ENTER HALF-BUS MODE
YES
STAY IN WHOLE BUS MODE

FIG. 9

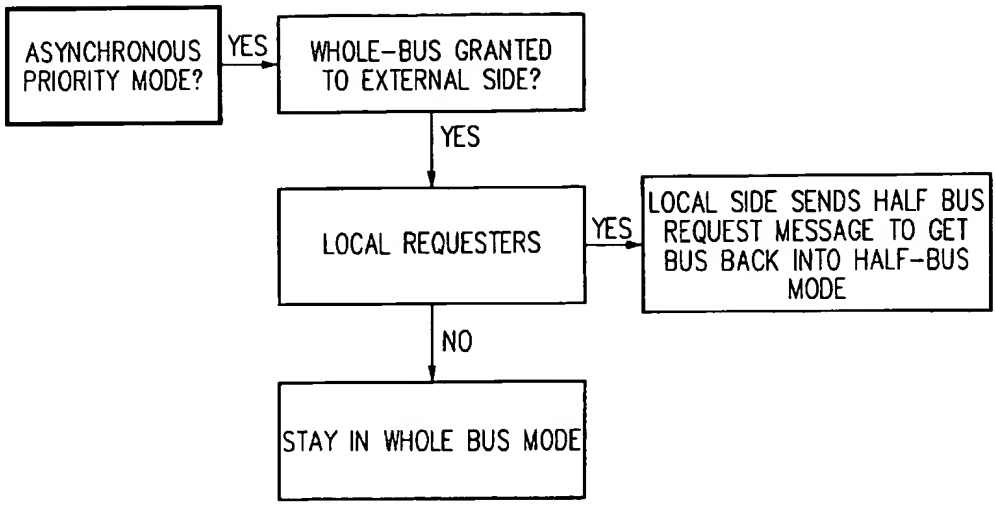

ASYNCHRONOUS PRIORITY MODE?
YES
WHOLE-BUS GRANTED TO EXTERNAL SIDE?
YES
LOCAL REQUESTERS
YES
LOCAL SIDE SENDS HALF BUS REQUEST MESSAGE TO GET BUS BACK INTO HALF-BUS MODE
NO
STAY IN WHOLE BUS MODE

FIG. 10

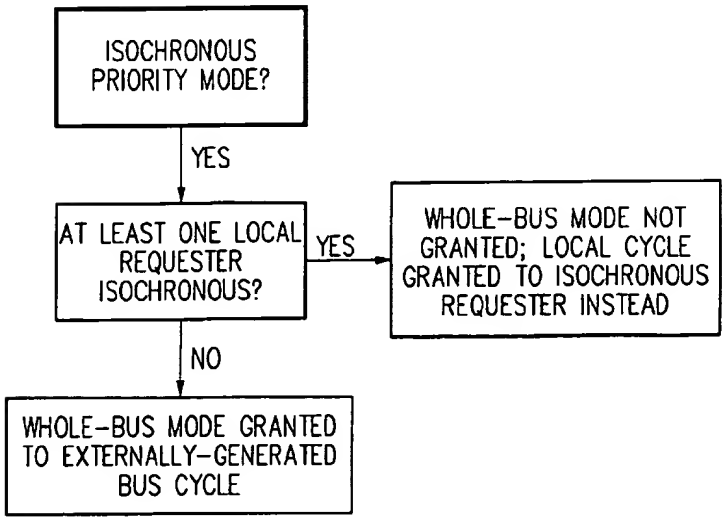

ISOCHRONOUS PRIORITY MODE?
YES
AT LEAST ONE LOCAL REQUESTER ISOCHRONOUS?
YES
WHOLE-BUS MODE NOT GRANTED; LOCAL CYCLE GRANTED TO ISOCHRONOUS REQUESTER INSTEAD
NO
WHOLE-BUS MODE GRANTED TO EXTERNALLY-GENERATED BUS CYCLE

FIG. 11

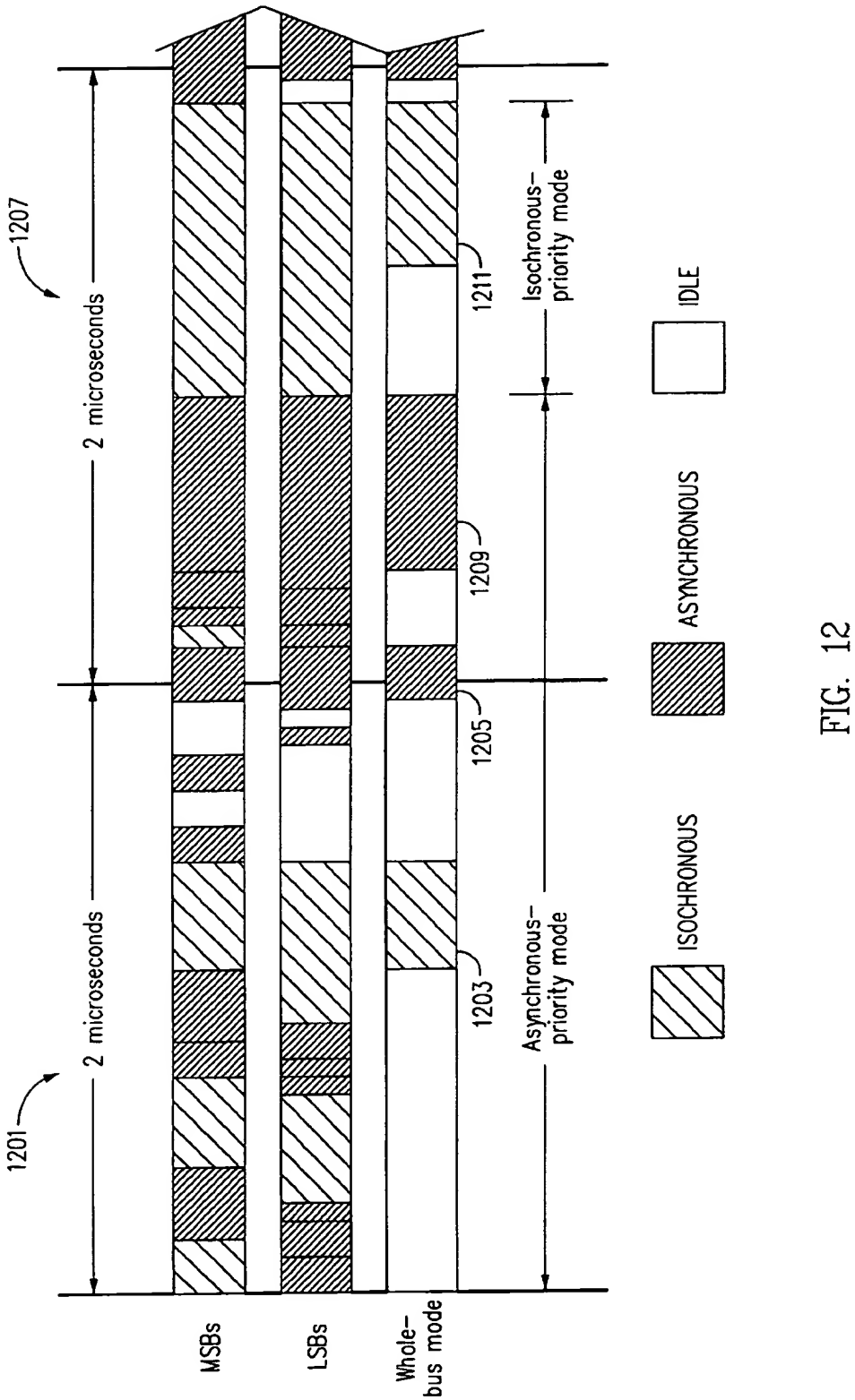

1207
2 microseconds
1201
2 microseconds
1211
1209
1205
1203
Isochronous-priority mode
Asynchronous-priority mode
IDLE
ASYNCHRONOUS
ISOCHRONOUS
MSBs
LSBs
Whole-bus mode

FIG. 12

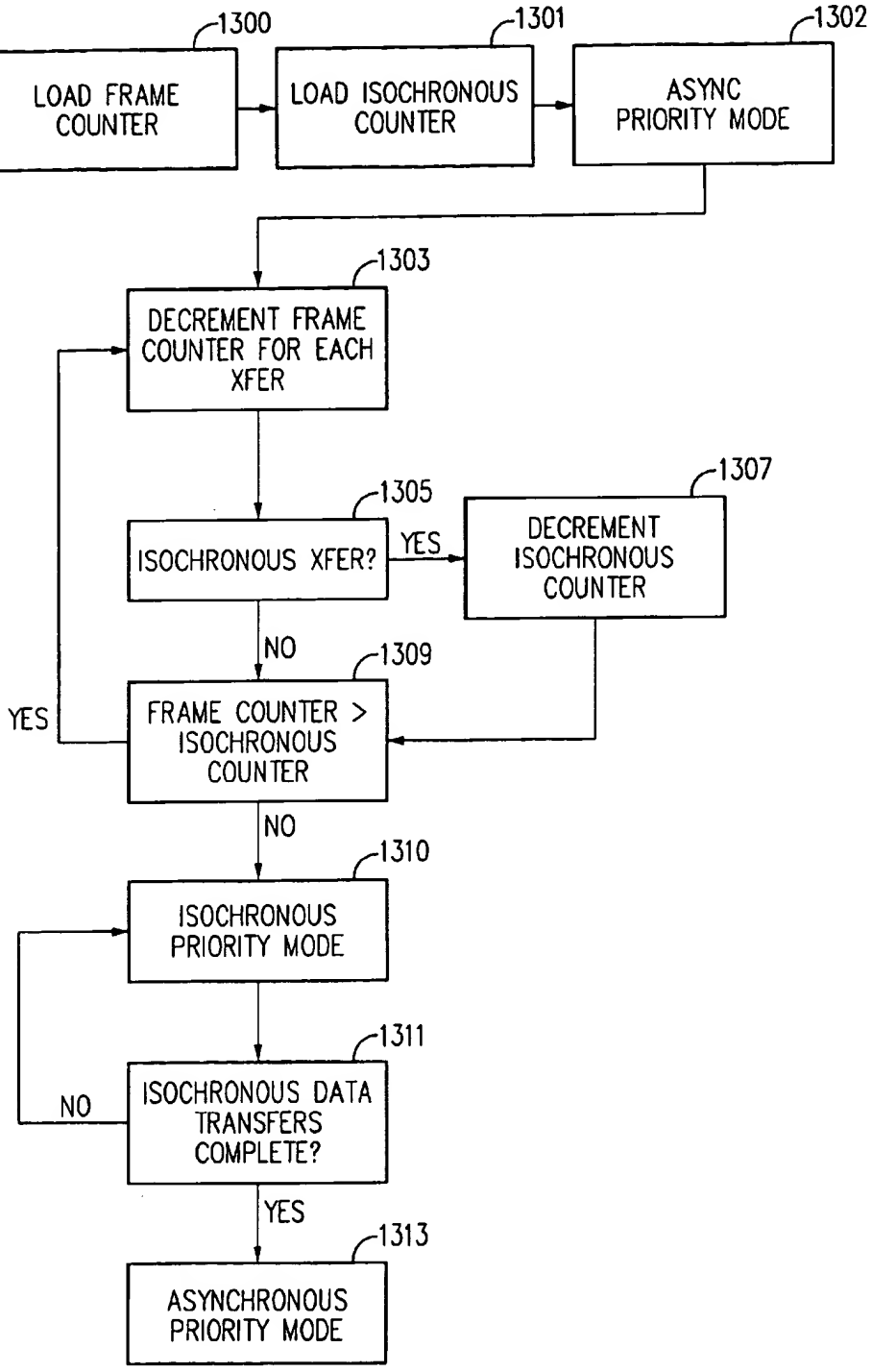

1300
LOAD FRAME COUNTER
1301
LOAD ISOCHRONOUS COUNTER
1302
ASYNC PRIORITY MODE
1303
DECREMENT FRAME COUNTER FOR EACH XFER
1305
ISOCHRONOUS XFER?
YES
1307
DECREMENT ISOCHRONOUS COUNTER
NO
1309
FRAME COUNTER > ISOCHRONOUS COUNTER
YES
NO
1310
ISOCHRONOUS PRIORITY MODE
1311
ISOCHRONOUS DATA TRANSFERS COMPLETE?
NO
YES
1313
ASYNCHRONOUS PRIORITY MODE

FIG. 13

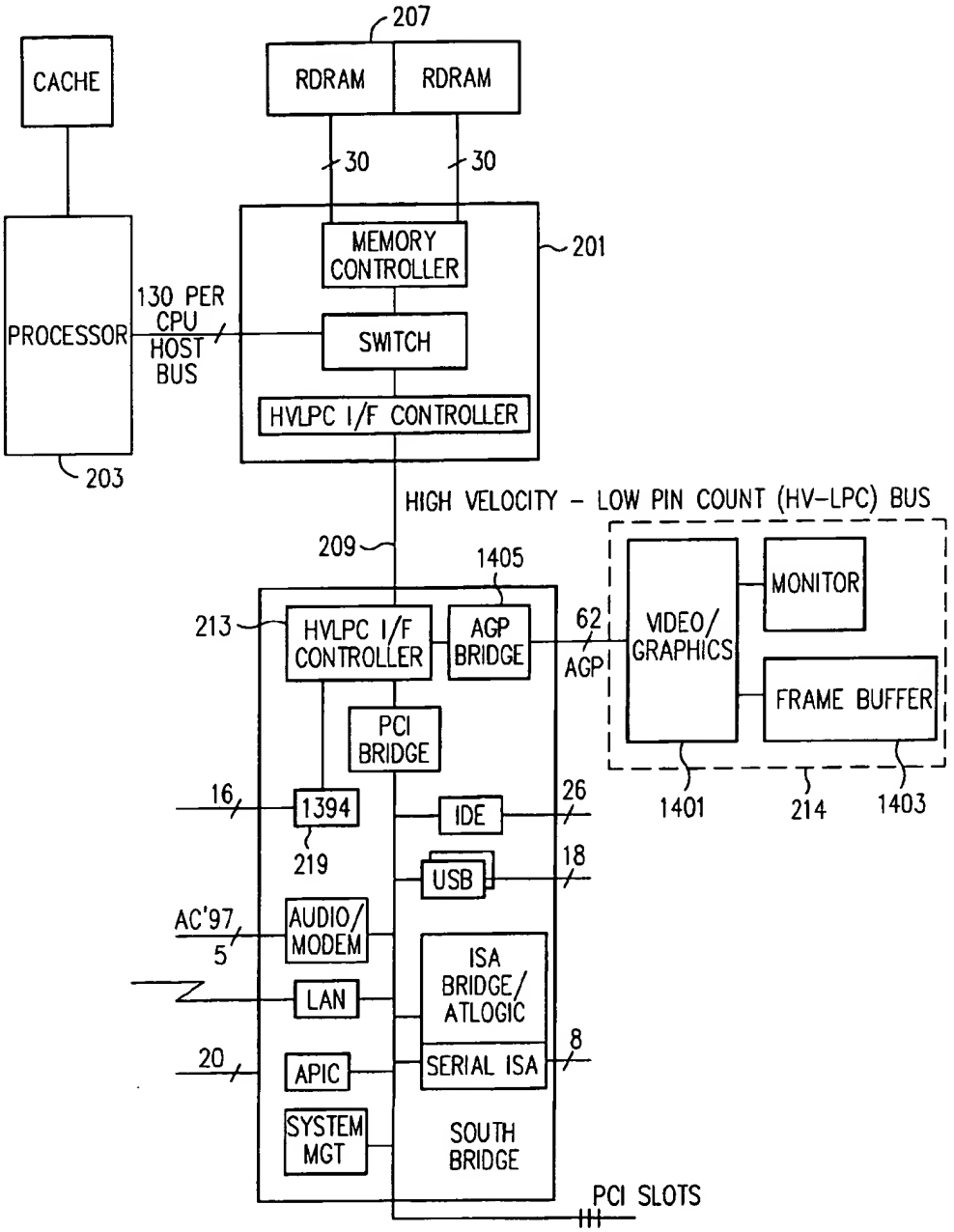

207
CACHE
RDRAM
RDRAM
30
30
MEMORY CONTROLLER
201
PROCESSOR
130 PER CPU HOST BUS
SWITCH
HVLPC I/F CONTROLLER
203
HIGH VELOCITY - LOW PIN COUNT (HV-LPC) BUS
209
1405
213
HVLPC I/F CONTROLLER
AGP BRIDGE
62
AGP
VIDEO/ GRAPHICS
MONITOR
FRAME BUFFER
PCI BRIDGE
16
1394
219
IDE
26
USB
18
AC'97
5
AUDIO/ MODEM
LAN
ISA BRIDGE/ ATLOGIC
20
APIC
SERIAL ISA
8
SYSTEM MGT
SOUTH BRIDGE
PCI SLOTS
1401
214
1403

FIG. 14

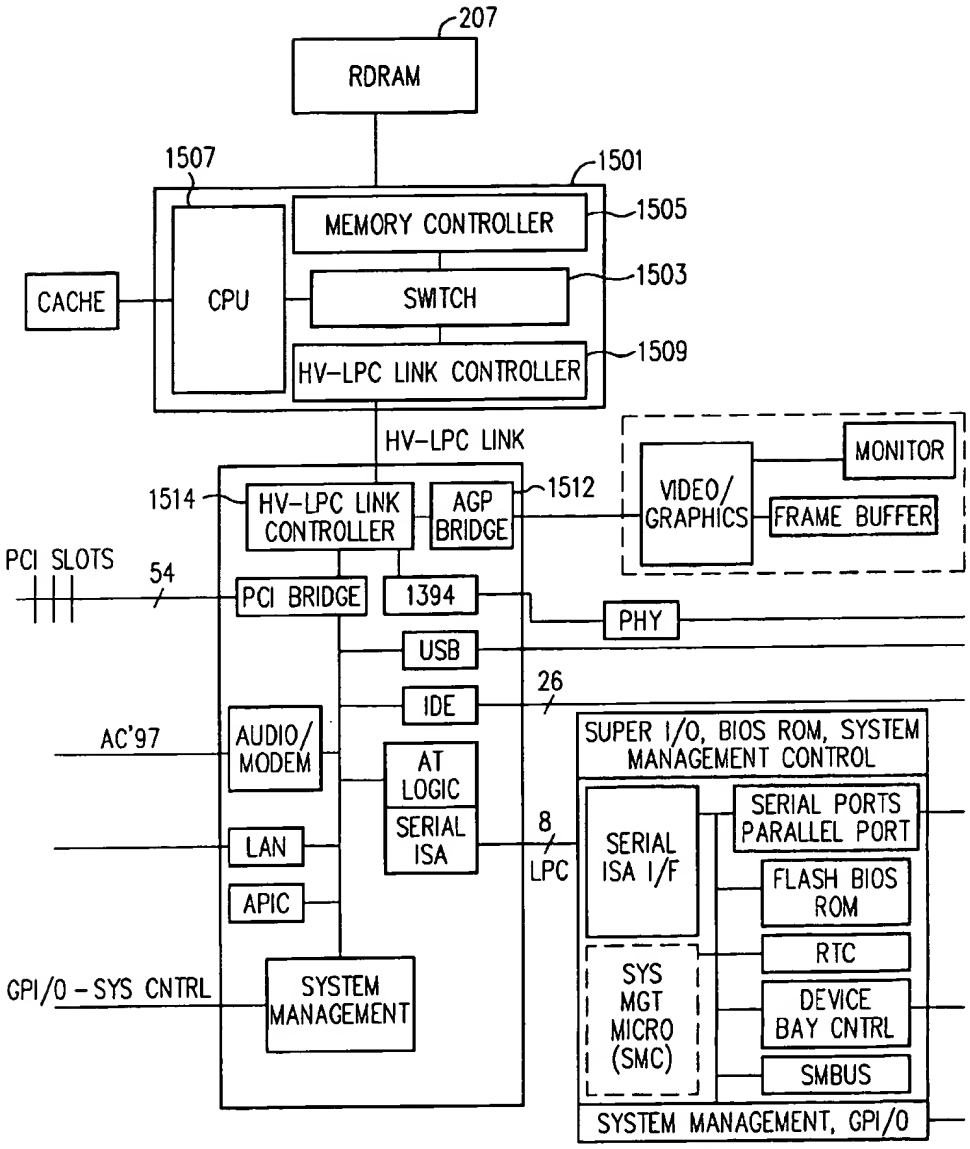

207
RDRAM
1507
1501
MEMORY CONTROLLER
1505
CACHE
CPU
SWITCH
1503
HV-LPC LINK CONTROLLER
1509
HV-LPC LINK
MONITOR
VIDEO/ GRAPHICS
FRAME BUFFER
1514
HV-LPC LINK CONTROLLER
AGP BRIDGE
1512
PCI SLOTS
54
PCI BRIDGE
1394
PHY
USB
IDE
26
AC'97
AUDIO/ MODEM
AT LOGIC
SERIAL ISA
LAN
APIC
8
LPC
SUPER I/O, BIOS ROM, SYSTEM MANAGEMENT CONTROL
SERIAL ISA I/F
SERIAL PORTS PARALLEL PORT
FLASH BIOS ROM
RTC
SYS MGT MICRO (SMC)
DEVICE BAY CNTRL
SMBUS
GPI/O - SYS CNTRL
SYSTEM MANAGEMENT
SYSTEM MANAGEMENT, GPI/O

FIG. 15

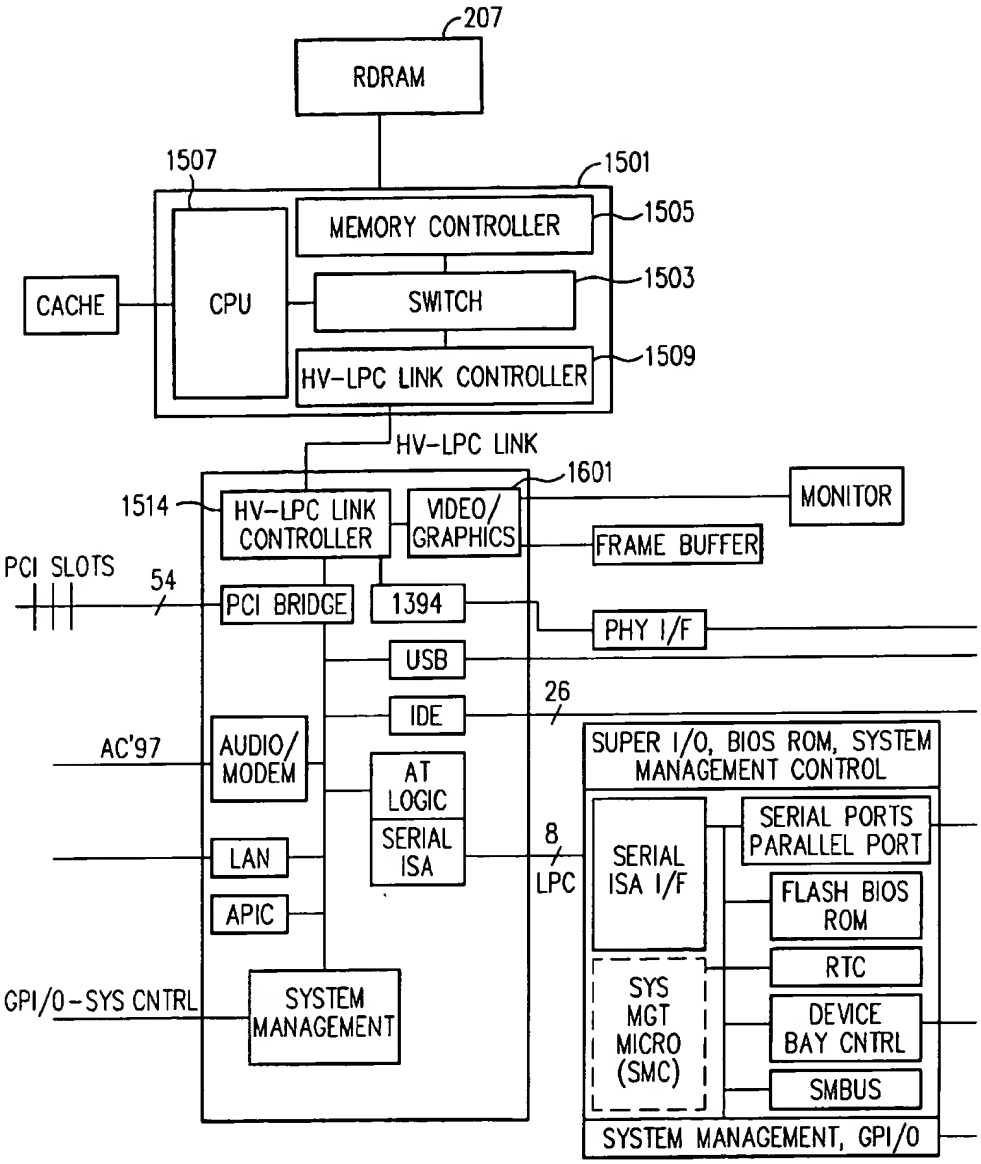

207
RDRAM
1507
1501
MEMORY CONTROLLER
1505
CACHE
CPU
SWITCH
1503
HV-LPC LINK CONTROLLER
1509
HV-LPC LINK
1601
1514
HV-LPC LINK CONTROLLER
VIDEO/ GRAPHICS
MONITOR
FRAME BUFFER
PCI SLOTS
54
PCI BRIDGE
1394
PHY I/F
USB
26
IDE
AC'97
AUDIO/ MODEM
AT LOGIC
SERIAL ISA
8
LPC
LAN
APIC
GPI/O-SYS CNTRL
SYSTEM MANAGEMENT
SUPER I/O, BIOS ROM, SYSTEM MANAGEMENT CONTROL
SERIAL ISA I/F
SERIAL PORTS PARALLEL PORT
FLASH BIOS ROM
RTC
SYS MGT MICRO (SMC)
DEVICE BAY CNTRL
SMBUS
SYSTEM MANAGEMENT, GPI/O

FIG. 16

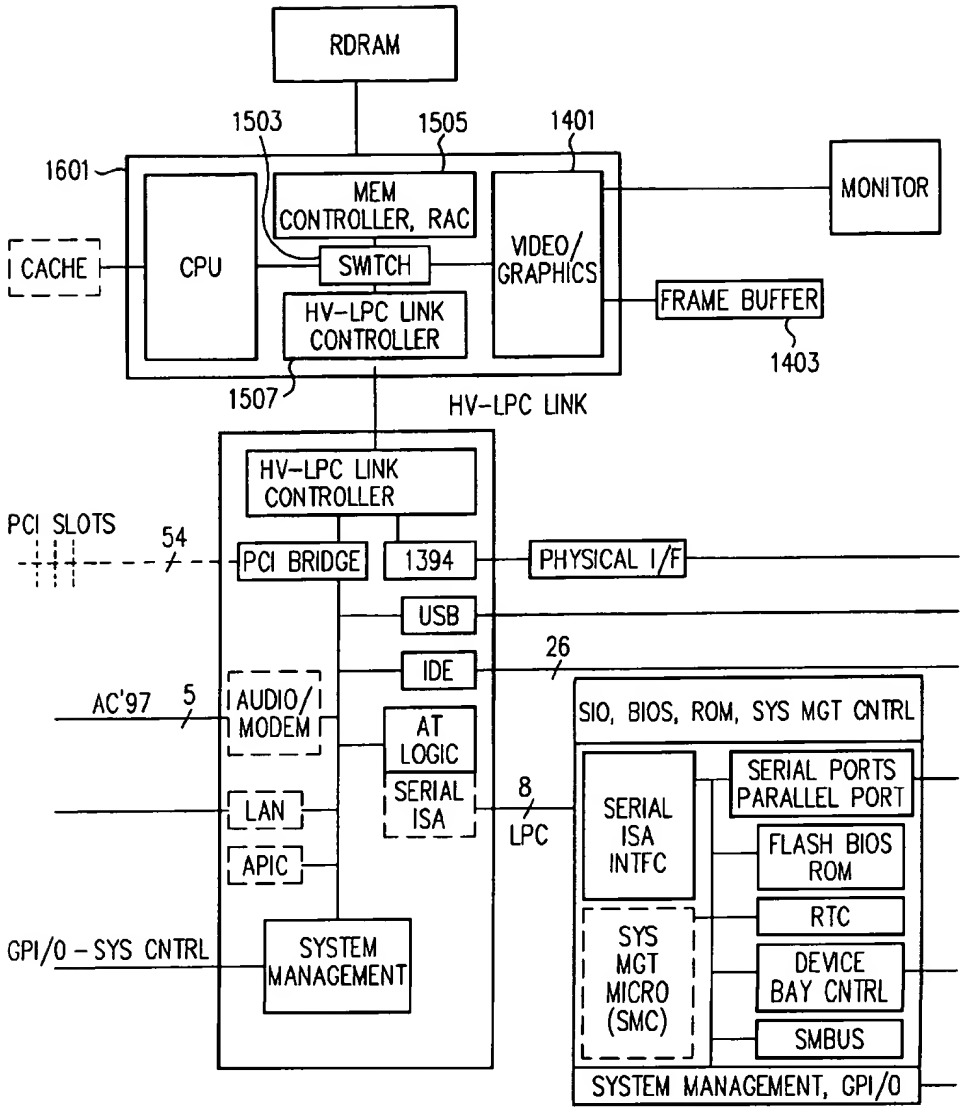

RDRAM
1503
1505
1401
1601
MEM CONTROLLER, RAC
MONITOR
CACHE
CPU
SWITCH
VIDEO/ GRAPHICS
FRAME BUFFER
HV-LPC LINK CONTROLLER
1403
1507
HV-LPC LINK
HV-LPC LINK CONTROLLER
PCI SLOTS
54
PCI BRIDGE
1394
PHYSICAL I/F
USB
26
IDE
AC'97
5
AUDIO/ MODEM
AT LOGIC
SIO, BIOS, ROM, SYS MGT CNTRL
SERIAL ISA
8
LPC
SERIAL ISA INTFC
SERIAL PORTS PARALLEL PORT
LAN
FLASH BIOS ROM
APIC
RTC
SYS MGT MICRO (SMC)
DEVICE BAY CNTRL
GPI/0 - SYS CNTRL
SYSTEM MANAGEMENT
SMBUS
SYSTEM MANAGEMENT, GPI/0

FIG. 17

COMMUNICATION LINK WITH ISOCHRONOUS AND ASYNCHRONOUS PRIORITY MODES COUPLING BRIDGE CIRCUITS IN A COMPUTER SYSTEM

CROSS REFERENCE TO RELATED APPLICATIONS

This application relates to co-pending application Ser. No. 09/098,876, filed Jun. 17, 1998, entitled WRITE ONLY BUS WITH WHOLE AND HALF BUS MODE OPERATION, by Larry Hewitt; application Ser. No. 09/099,227, filed Jun. 17, 1998, entitled METHOD OF MODE CONTROL IN A BUS OPTIMIZED FOR PERSONAL COMPUTER TRAFFIC, by Larry Hewitt, now U.S. Pat. No. 6,032,211; co-pending application Ser. No.09/098,854, filed Jun. 17, 1998, entitled A BUS OPTIMIZED FOR PERSONAL COMPUTER TRAFFIC, by Larry Hewitt and Dale E. Gulick; and co-pending application Ser. No. 09/099,228, filed Jun. 17, 1998, entitled CPU-NORTH BRIDGE INTEGRATION UTILIZING AN INTERCONNECTION BUS PROVIDING A HIGH SPEED-LOW PIN COUNT LINK, by Dale E. Gulick and Larry Hewitt, which applications are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates to computer systems and more particularly to a computer system including a bus having a relatively high bandwidth and a relatively low pin count.

2. Description of the Related Art

Traditional personal computer architectures partition the computer system into the various blocks as shown in FIG. **1**. The central feature of this prior art architecture is the use of the Peripheral Component Interface (PCI) bus **101** as the connection between the "north bridge" integrated circuit **103** and the "south bridge" integrated circuit **105**. The north bridge functions generally as a switch connecting CPU **107**, a graphics bus **109** such as the Advanced Graphics Port (AGP) bus, and the PCI bus to main memory **111**. The memory controller function is also located in the north bridge.

The south bridge generally provides the interface to the input/output (I/O) portion of the system with the exception of video output as illustrated in the prior art computer architecture shown of FIG. **1**. Specifically, the south bridge **105** provides a bridge between the PCI bus and legacy PC-AT (Advanced Technology) logic. The south bridge also provides a bridge to the legacy ISA bus **115**, the Integrated Device Electronics (IDE) disk interface **117** and the Universal Serial Bus (USB) **119**. The various busses and devices shown in FIG. **1** are conventional and are not described further herein unless necessary for an understanding of the present invention.

In the illustrated prior art architecture, the PCI bus between the north bridge and the south bridge also functions as the interconnect bus for many add-in functions. That results in a significant number of pins on the north bridge circuit **103** and the south bridge circuit **105** to account for the add-in functions. That also results in a lack of determinism in the system because any function on the PCI bus can become master of the bus and tie up the bus. Ideally, communication between the CPU and the resources in the south bridge, or between the resources in or coupled to the south bridge and system memory **111** should be deterministic in the sense of knowing what throughput is available for a particular transfer and the latency that is involved for that transfer.

It would be desirable to have a deterministic system for the major interconnect bus and in addition to reduce the pressure for additional pins on the integrated circuits making up the computer system.

SUMMARY OF THE INVENTION

Accordingly, the invention provides a computer architecture that includes a new interconnect bus between the north bridge and the south bridge integrated circuits. As part of the new architecture, the PCI bus no longer connects to the north bridge integrated circuit.

In one embodiment, the invention provides a computer system that includes a first processor integrated circuit. A first bridge integrated circuit is coupled to the processor via a host bus. The computer system includes an interconnection bus that couples the first bridge circuit and a second bridge circuit. The interconnection bus provides a first transfer mode for asynchronous data and a second transfer mode for isochronous data. The interconnection bus provides for a maximum latency and guarantees a minimum throughput for isochronous and asynchronous data.

In another embodiment a method of communicating information in a computer system is provided. The computer system includes a processor coupled to a first (north) bridge integrated circuit by a host bus and an interconnection bus connecting a second (south) bridge integrated circuit to the first (north) bridge integrated circuit. The interconnection bus provides a first transfer mode for asynchronous data and a second transfer mode for isochronous data on the interconnection bus. Data is transmitted between the first bridge integrated circuit and the second bridge integrated circuit in either the first or the second transfer modes via the interconnection bus.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

FIG. **1** depicts a prior art personal computer architecture.

FIG. **2** depicts a first embodiment of the invention showing a point-to-point interconnect between the north bridge and the south bridge.

FIG. **3** shows further details of the interconnect bus and the bus interfaces.

FIG. **4** illustrates a flowchart for determining when to enter isochronous priority mode.

FIG. **5** illustrates the link layer, protocol layer and physical layer of the bus.

FIG. **6** illustrates one channel of the link layer.

FIG. **7** illustrates the whole and half bus modes on the bus according to one embodiment of the invention.

FIG. **8** is a flow diagram illustrating when a local controller should grant whole-bus mode to an external controller when the bus is in asynchronous priority mode.

FIG. **9** is a flow diagram illustrating when a local controller should stay in whole-bus mode at the end of a bus cycle when the bus is in asynchronous priority mode.

FIG. **10** is a flow diagram illustrating when a local controller stays in whole-bus mode after a whole-bus cycle is granted to the external side.

FIG. **11** is a flow diagram illustrating when to grant whole bus mode while the bus is in isochronous priority mode.

FIG. **12** shows a traffic example including isochronous and asynchronous data and whole and half-bus modes.

FIG. 13 shows a flow chart for determining when to enter isochronous priority mode.

FIG. 14 shows a computer system in which the graphics subsystem is coupled to the south bridge.

FIG. 15 shows a computer system in which the processor and the north bridge have been integrated into a single integrated circuit which is coupled to the south bridge via the interconnection bus.

FIG. 16 shows a computer system having an integrated processor and memory controller, which is coupled via the interconnection bus to the south bridge which has an integrated graphics controller function.

FIG. 17 shows a computer system having an integrated processor and memory controller and graphics controller, which is coupled via the interconnection bus to the south bridge.

DESCRIPTION OF THE PREFERRED EMBODIMENT(S)

Referring to FIG. 2 a computer system according to one embodiment of the present invention includes north bridge integrated circuit **201** coupled between processor integrated circuit **203**, system memory **207**, graphics subsystem **214** and south bridge integrated circuit **211**. The processor **203** may include both a processor core and other functions such as cache memory. In the illustrated embodiment, the system (or main) memory shown in FIG. 2 utilizes Rambus dynamic random access memory (RDRAM) to provide fast DRAM access. The graphics subsystem is shown coupled to the north bridge integrated circuit via an accelerated graphics processor (AGP) bus. The graphics subsystem includes a graphics controller circuit **220**, monitor **222** and frame buffer **224**. Graphics controller circuits are known in the art. One example is the Intel 740 Graphics Accelerator integrated circuit. Other graphics buses beside the AGP may of course be utilized.

The north bridge integrated circuit **201** provides a switching function connecting processor **203**, graphics bus **205**, memory **207**, and interconnect bus **209**. The interconnect bus replaces the PCI bus shown in the prior art computer architecture. The interconnect bus **209** has a high speed relative to the PCI bus and a relatively low pin count. The interconnect bus is also referred to herein as the high velocity—low pin count (HV-LPCP). System memory **207** and AGP bus **205** are coupled to the switch through memory controller **216** switch and AGP bridge circuit **218** respectively. Interconnect bus interface bus interface **210** (HV-LPC I/F) provides an interface to interconnect bus **209** for those circuits coupled to switch **212**.

The south bridge integrated circuit **211** provides a bridge between interconnect bus **209** and PCI bus **215** through PCI bridge **217**. The south bridge **211** also provides a bridge to the **1394** bus through the **1394** interface circuit **219**. Other functions that are also accessible through the PCI bus include IDE, Universal Serial Bus (USB), Industry Standard Architecture (ISA) bridge and Advanced Programmable Interrupt Controller (APIC).

One main feature of the architecture shown in FIG. 2 is that bus **209** provides a guaranteed minimum bandwidth and a maximum latency to data transferred over the bus. That is accomplished, as described further herein, by transferring data over the interconnect bus in frames, with each frame guaranteeing a portion of the frame for isochronous data and a portion of the frame for asynchronous data.

Guaranteeing maximum latency for various channels connected to the bus is becoming more important as isochronous data streams are being conveyed between the south bridge and main memory. One source of such isochronous data is the IEEE **1394** bus.

Another feature of the architecture shown in FIG. 2 is that the PCI bus function has moved to the south bridge. The HV-LPC link **209** provides the PCI-resident functions all of the bandwidth that existed in the old architecture, and also provides additional capacity for new functions such as the **1394** bus. Note that the **1394** bus can cause as big a load on the system as the PCI bus. That implies that the interconnect bus must be high speed relative to at least the PCI bus.

One way to achieve a high speed bus is to utilize a point-to-point bus in which only two devices are present on the bus. A point-to-point bus can inherently run at higher speeds than a multi-drop bus such as the PCI bus since a point-to-point link has reduced electrical loading and reduced noise caused by reflections at tap points such as connectors. It is possible to provide a point to point link that operates at 25 times the speed of the PCI bus. Given this, the 32-bit wide PCI bus can be replaced by a 16-bit (or even an 8-bit link) while adding significant bandwidth.

The interconnect bus **209** will now be described in greater detail. Referring to FIG. **3** bus **209** which connects a first module (which may be a processor module on a north bridge as described further herein) and the interface module, includes bi-directional data portions **307** and **309**. In one embodiment, each data portion contains one byte (8 bits) of data. However, the number of bits on the data bus may be of size (2^n-1:0), where n is an integer>0 that meets the throughput requirements of the system. Thus, a minimum implementation may have one data bit in each direction. In the illustrated embodiment, n equals 4, with each data portion having one byte.

Referring to FIG. 2 and FIG. 3, the bus interface **210** in north bridge **201** is arbitrarily designated side A and the bus interface **213** in south bridge **211** is designated as side B. Bus **209** includes a unidirectional clock line CLKB2A and a unidirectional control line CTLB2A provided by bus interface **210** to bus interface **213**. The "B2A" designation indicates that the signal is an output of side B and an input to side A. Bus **209** also includes a second unidirectional clock line CLKA2B and a second unidirectional control line CTLA2B, which are provided by side A to side B. The "A2B" designation indicates that the signal is an output of side A and an input to side B. The protocol uses clock-forwarding technology to reliably synchronize source data to a clock. CLKA2B and CLKB2A are preferably derived from the same source such that they are the same frequency and they do not drift.

Each bus interface side includes a transmit controller and a receive controller. Data always flows from the transmit controller on one side to the receive controller on the other side. Thus, side A bus interface **210** in north bridge **201** includes transmit controller **315** and a receive controller **313**. Side B bus interface **213** in south bridge **211** includes transmit controller **311** and receive controller **317**. The two bi-directional portions **307** and **309** of the data bus are shown with arrows indicating their default direction of transfer. Data bus portion **307** transmits data in a default mode of operation, from transmit controller **311** to receive controller **313** (from side B to side A) synchronous with CLKB2A. Data portion **309** of the data bus is dedicated, in a default mode of operation to transmit data from transmit controller **315** to receive controller **317** (side A to side B), synchronously with CLKA2B. In the illustrated embodiment, bus **209** is a point to point bus physically

connecting precisely two integrated circuits. In that way, transfer speed across the bus may be maximized. However, some embodiments of the bus may connect more than two integrated circuits to the bus.

The side A transmit controller delivers clock CLKA2B to the side B receive controller, and the side B transmit controller delivers clock CLKB2A to the side A receive controller. Thus, the side A transmit controller and side B receive controller are included in the same time domain, called time domain A, and the side B transmit controller and the side A receive controller are included in the same time domain, called time domain B.

In the illustrated embodiment, both side A and side B initialize to a mode of 16 bits wide and 800 megahertz data rate. The data rate is based on a clock (CLK) (meaning CLKA2B and CLKB2A) rate of 400 megahertz and the data is provided on each edge of the clock. So, if clock is 400 megahertz, this represents 800 million edges per second. Address phases and data phases of bus cycles transfer information at each edge of CLK. Thus, the edge rate of CLK specifies the maximum theoretical bandwidth of the bus rather than the cycle rate.

Two kinds of traffic occur over bus **209**: bus cycles and messages. The greatest amount of bandwidth is used by bus cycles, which are transfers of blocks of addressing information or addressing information and associated data sent from one link layer to the link layer on the other side. The addressing information determines where in the integrated circuit a particular access is targeted. A bus cycle may be a long or short bus cycle. A long bus cycle may utilize several hundred nanoseconds of bus time while a short bus cycle uses less than, e.g., 20 or 30 nanoseconds. The second type of bus traffic is messages, which are used to send protocol information across the link. In one embodiment, messages are aligned to the rising edge of CLK and consume one CLK cycle and can occur at any time, including in the middle of bus cycles. In other embodiments, messages may last more than one cycle.

In order to provide the necessary isochronous bandwidth on the bus for isochronous streams, the isochronous streams are guaranteed a specified amount of bandwidth during each frame on the bus. In the illustrated embodiment, frames are 2 microseconds in length. Referring to FIG. **4**, a flowchart illustrates one method of guaranteeing sufficient isochronous bandwidth according to the present invention. Initially during the frame, asynchronous transfers are granted priority over isochronous transfers (to minimize the latency of the asynchronous transfers), which is called asynchronous priority mode. Thus, at the start of each new frame **401**, the bus enters asynchronous priority mode at **403**. In asynchronous priority mode, asynchronous transfers will be transferred instead of isochronous transfers if there are asynchronous requesters for the bus. However, isochronous data may be transferred when the bus is in asynchronous priority mode when there is no asynchronous data to transfer. If the frame is not over at **405**, step **407** determines whether it is necessary to start transferring only isochronous data (isochronous priority mode) to guarantee sufficient bandwidth is provided for isochronous data. If so, the bus switches to isochronous priority mode in step **409**. The bus stays in isochronous priority mode until all isochronous transfers are complete. There is preferably some cushion factored into the isochronous determination in step **407** so that there is at least some time left in the particular frame after all isochronous data is transferred. After transferring all isochronous data, the bus returns to asynchronous priority mode at **413** until the frame is over at **415**. It is possible for the bus to stay in asynchronous priority mode the entire frame without switching to isochronous priority mode as is described further herein.

The bus **209** is very useful in situations-where high-bandwidth asynchronous traffic must be mixed with isochronous traffic. The bus protocol assumes (1) that system performance is adversely affected by the latency of asynchronous traffic, (2) asynchronous traffic can be delayed indefinitely without adversely affecting real-time data streams, (3) isochronous traffic must be guaranteed a specified amount of bandwidth and worst-case latency, and (4) as long as the bandwidth and latency requirements for isochronous traffic are met, then the latency between their requests and the transfer of the data has no adverse affect on system performance.

The hardware on each side of the bus includes a physical layer, a protocol layer, and a link layer. The protocol layers for both sides of the bus include the same hardware elements. In this way, the bus is symmetrical with no centralized resources (as opposed to, for example, the PCI bus arbiter which in prior art systems was typically located in the north bridge of the PCI bus and arbitrates for all masters).

Referring to FIG. **5**, the bus includes link layer **501**, protocol layer **503**, and physical layer **505**. The physical layer will depend on such factors as the frequency of the bus, the number of devices on the bus, the length of the bus, as is known to those of skill in the art. The specification for the physical layer and the protocol layer is generally device independent, except for variations of the bus width and frequency. The specification for the link layer varies based on the requirements of the device.

In the illustrated embodiment, there may be one channel in the link layer for each functional unit connected to the bus. For example, referring again to FIG. **2**, north bridge **201** may include three channels which are coupled to the bus interface **210** through switch **212**. One channel is for processor **203**, one for main memory **207**, and one for the graphics subsystem **214**. The corresponding south bridge **213** may include two channels: one for the **1394** interface and one for one for the PCI bus. In addition, additional channels may be provided for the USB because it has isochronous data, and/or for an IDE interface, and one for an expansion bus interface. The link layer also includes an arbiter **507** to determine the source of the next locally-generated bus cycle since there are typically multiple asynchronous and isochronous sources. The arbiters guarantee bandwidth to isochronous streams (within a maximum latency) while minimizing latency to asynchronous accesses.

Referring to FIG. **6** a typical channel in the link layer is shown. Each channel in the link layer includes FIFOs and queues of addressing information and data that have been sent across the bus or that will be sent across the bus. Thus, in FIG. **6** asynchronous transmit FIFO(s) **601** store asynchronous data that will be sent across the bus while asynchronous receive FIFO(s) **603** store asynchronous data received from the protocol layer. The channel shown in FIG. **6** also includes queues of asynchronous read requests **605** and access requests **607**. Isochronous transmit and receive FIFO(s) **609** and **611** respectively store isochronous information for transmitting and store isochronous information that has been received.

The link layer hardware, unlike the protocol layer, is specific to the requirements of the local integrated circuit on which the link layer is implemented. FIFOs and queues are designed to the specific requirements of the channels being

serviced by the link. Generally, the FIFOs of each channel are optional, based on the channel requirements. For example, one would not expect any isochronous transmit or receive FIFOs for the south bridge's PCI block, since the PCI protocol does not allow for guaranteed isochronous data transfers (although, nothing prevents a designer from including these for the PCI bus channel). The 1394 FIFO, on the other hand would expect to have both isochronous transmit and receive FIFOs. It is also possible in certain situations that there may be only a single receive FIFO and potentially no transmit FIFOs at all.

FIFOs utilized in the link layer are designed with the following considerations in mind. The FIFO may contain isochronous or asynchronous data; the FIFO may transmit data to the bus or receive data from the bus; the FIFO has a predetermined size (in bytes); the FIFO may be a master (controlled by local timing) or a slave (responding only to accesses); if a FIFO is a slave, then design considerations include what causes the slave FIFO to be loaded. The FIFO target address may be static or it may increment with each byte, in which case it will be reloaded periodically.

The size of the isochronous FIFOs can be determined according to the following equation: (maximum required bandwidth in bytes per second)x(frame time in seconds)x2. For example, to support a 600 megabyte per second data stream, the FIFO would be required to be 2400 bytes deep.

The processor module's system memory channel has special requirements, since multiple streams from the interface module may attempt to access it. It requires several isochronous transmit and receive FIFOs and, potentially, multiple asynchronous FIFOs.

There are special design considerations for the system memory channel because it is required to connect to widely varying interface module channels on the other side of the bus. So the system memory channel may include a 2× bandwidth RAM and an array of programmable head and tail pointers for various FIFOs. A typical implementation may include a 4K byte block of SRAM and 16 head and tail pointer pairs. Software configures these based on the channel requirements on the other side of the link.

As previously mentioned a bus cycle is defined as a block transfer of either addressing information by itself or addressing information followed by data. The address phase of a bus cycle transmits the addressing bytes over the link and the data phase, if there is one, transmits the associated data bytes over the link. The bus is write only in that reads are accomplished by (1) sending a read request from a first side to the second side, after which, (2) the read data is retrieved and, in a separate cycle, sent from the second side to the requesting side. Bus cycles are granted to either asynchronous or isochronous requesters. Requesters are link-layer devices that are currently requesting use of the link for a bus cycle.

The address phase occurs at the beginning of each bus cycle and typically includes the type of bus cycle being transmitted, the number of bytes to be transmitted in the bus cycle, bus-specific cycle type and address and whether both halves of the bus are requested for the transfer. There are several types of cycles in one embodiment as illustrated in Table 1.

TABLE 1

Cycle Types

Cycle type	Description
Asynchronous read request	Send request to read data from one side to another
Asynchronous read response	Send requested data back across the bus
Asynchronous write request	Send address and data from one side to another
Asynchronous write response	Acknowledgment back to initiator of write that data has been received
Isochronous read request	Send request to read data from one side to another
Isochronous read response	Send requested data back across the bus
Isochronous write	Send address and data from one side to another

In a default mode of operation, the data bus of the illustrated embodiment is split in half, with the 8 least significant bits (LSBs) of each 16-bit block controlled by the side A time domain and the 8 most significant bits (MSBs) of each 16-bit block controlled by the side B time domain. However, the bus protocol allows one side to utilize both halves of the bus to transmit data. The "local half-bus" refers to the half of the bus that is defaulted to be owned by the local transmit controller on an integrated circuit (LSBs for side A and MSBs for side B). The "external half-bus" refers to the other half of the bus, the half that is defaulted to be owned by the transmit controller on the other side of the link.

Thus, the protocol layer includes an arbiter (**509** in FIG. **5**), called the whole-bus mode arbiter, for the local half-bus to determine if it will be used to transmit data (default mode) or if it will be used to receive external data. Half-bus mode refers to the default state of the link, in which side A controls the LSBs and side B controls the MSBs in the illustrated embodiment. Whole-bus mode refers to the state in which both halves of the data bus have been granted to a transmit controller. The whole-bus mode arbiter only determines when to enter whole-bus mode. Arbitration for a locally-initiated bus cycle is handled via the link layer by the local request arbiter.

Short bus cycles (e.g., accesses that consume 20 nanoseconds of bus time or less, while only using half the data bus) are completed solely on the local half-bus. Long accesses (e.g., longer than 30 nanoseconds) may utilize several hundred nanoseconds of bus time, and start by sending data over local half-bus. While data is being transferred, arbitration for the external half-bus occurs (determining if there are local requesters), and then once the external half-bus is granted, the whole bus is utilized for the remainder of the bus cycle's transfers. Short bus cycles are allowed to complete without being halted. Long bus cycles can be halted for higher-priority traffic and then re-started at a later time.

Referring to FIG. **7**, typical traffic over the bus **109** in whole and half-bus mode is illustrated. The side B half-bus, transferring data from side B to side A is shown at **702**. The side A half-bus, transferring data from side A to side B is shown at **704**. In the illustrated embodiment, all short bus cycles (e.g., accesses that consume 30 nanoseconds of bus time or less) utilize only half-bus mode. Thus, the short cycle shown at time period **701** which transfers data from side B to side A, is completed solely on the local half-bus. During time period **703**, both halves of the bus are idle as indicated by the "I". At the beginning of time period **705**, side A begins a long access, which as described, may utilize

several hundred nanoseconds of bus time. Initially, during the long access, A transfers data only on the local half-bus. However, while data is being transferred, side B's whole-bus mode arbiter determines whether to grant its half-bus to the external controller during time period **707**. Since side B is not using its local half-bus at this time, it grants its use to side A and the long cycle is completed in whole bus mode during time period **709**. Thus, the bus is able to exploit idle time by granting use of an idle half-bus to a requesting half bus with a long transfer. Once the half-bus is granted by side B, the whole bus is utilized for the remainder of side A's bus cycle.

During time period **713**, side A executes another long cycle transferring data from side A to side B. A side B long cycle begins during time period **715**. During time period **717**, side A's long cycle **713** has been completed and a short cycle from side A to side B occurs. Once that short cycle completes, side A is available to side B. The arbitration for side A's half-bus occurs during time period **719**. Side A grants its half-bus to side B which causes the bus to switch to whole bus mode. The cycle completes in whole bus mode during time period **721**. Once the bus cycle is complete, the bus again enters half-bus mode at **723**.

In certain embodiments, all initiated bus cycles over the bus are allowed to complete without interruption. If, during the transmission of data, the bus enters whole-bus mode, then it will stay in that mode until the bus cycle is completed. Both side A and side B keep track of the number of bytes being transmitted and when the bus cycle is complete, both sides resume half-bus mode as shown at **711**. In other embodiments, as described further herein, bus cycles may be interrupted and entering and leaving whole-bus mode differs according to whether the bus is operating in isochronous or asynchronous priority mode.

Maximum bandwidth requirements are specified for each isochronous stream in terms of bytes per frame. The sum of the isochronous-stream maximum-bandwidth requirements must be less than the theoretical total bandwidth. However, as a matter of practicality, the higher the percentage bandwidth of isochronous streams, the greater the probability that asynchronous traffic will incur latency. In the design of a balanced system, one expects the sum of typical asynchronous bandwidth and maximum isochronous bandwidth to be less than about 60 to 80 percent of the theoretical maximum bandwidth. If that is done, then average latency for asynchronous cycles will be minimized.

Isochronous requesters should not send more bytes across the link, during a frame, than the programmed maximum bandwidth for that requester. Hardware may be implemented in the link layer to ensure that isochronous requesters comply with that requirement.

As discussed, in one embodiment, bus traffic is grouped into 2-microsecond frames. Two counters associated with frames are used in the local request arbitration logic (**507** in FIG. **5**). They are the elapsed frame counter, which is used to specify how much bandwidth remains in the frame, and the isochronous byte counter, which is used to specify how much isochronous bandwidth remains to be transferred in the frame.

The elapsed frame counter starts, at the beginning of each frame, at a value equal to the number of bytes that can be transferred across the frame (product of the width of the bus in bytes and the number of clock edges in the frame). For example, in a 16-bit, 800 megahertz implementation (data rate), the value of the counter would start out at (1600 bytes per microsecond)x(2 microseconds per frame)=3200. It counts down to zero over the course of the frame, reloads, counts down again, and so forth. When the elapsed-frame counter reaches zero, a new frame is defined to be started. This is true for both sides of the link. When the elapsed frame counter reaches zero, in certain embodiments a new frame (NewFrame) message to is sent across the link to side B, which causes B's elapsed-frame counter to reset.

The isochronous byte counter starts, at the beginning of each frame, at a value equal to the number of isochronous bytes that must be guaranteed to be transferred during the frame. It decrements with each isochronous byte transferred. It is programmed to be slightly higher than the actual maximum isochronous bandwidth of a frame. Shortly after the beginning of each frame, all the isochronous streams make their requests to send data across the bus during the next frame. The requests may be made in the illustrated embodiment within a predetermined time period after the frame starts.

Initially during a frame, asynchronous transfers are granted priority over isochronous transfers (to minimize the latency of the asynchronous transfers), (asynchronous priority mode). However, counter(s) track of how much isochronous traffic passes during the frame and if the isochronous streams are in danger of running out of the required bandwidth for the frame, arbitration priority switches to the isochronous traffic, (isochronous-priority mode). In that way, a minimum amount of isochronous bandwidth can be guaranteed while minimizing latencies for asynchronous transfer requests. After all the isochronous bus cycles for the frame are complete (which occurs before the end of the frame), the priority switches back to the asynchronous traffic.

The bus protocol includes two arbiters: the whole-bus mode arbiter **509** in the protocol layer and the local-request arbiter **507** in the link layer that determines which local requester will next be granted the local half-bus. The local request arbiter operates as follows.

At the beginning of each frame, all isochronous streams that will require bandwidth during the frame request the link from the local-request arbiter. They continue to request the link until they have completed all of their bus cycles for the frame.

It is expected that the total requested bandwidth for the vast majority of frames will be well under 100% and the asynchronous transfers will be granted highest priority for the entire frame. Thus, as described above, asynchronous CPU accesses will most often be granted higher priority than isochronous transfers and therefore incur reduced average latency. In situations where large blocks of bulk asynchronous data are being transferred across the link (for example, from the PCI bus), then the isochronous transfers will tend to come at the end of the frame, after the priority has switched to isochronous bus cycles. In this case, CPU latency will tend to increase as the CPU loses priority to the isochronous bus cycles and contends with the bulk asynchronous transfers.

For each frame, the link either stays in asynchronous priority mode for the entire frame or (1) starts in asynchronous priority mode, (2) transitions to isochronous priority mode during the frame, and (3) then transitions again to asynchronous priority mode before the end of the frame as illustrated in FIG. **4**.

The rules for the local-request arbiter are very simple: asynchronous requesters are higher priority than isochronous requesters during asynchronous-priority mode and only isochronous requesters are granted bus cycles during

isochronous-priority mode. The arbitration method for the group of asynchronous requesters is not limited other than it is required to be fair and to not cause deadlock situations. The arbitration scheme for the group of isochronous requesters may utilize a fixed priority scheme.

The rules for whole-bus mode arbitration and operation and for entering and exiting whole-bus mode in one embodiment are described below and illustrated in FIGS. **8–11**. The change to isochronous priority guarantees that all isochronous bus cycles will complete before the end of the frame.

If the local half-bus is in use for a locally-generated cycle, then arbitration will not take place until that bus cycle is ended. The end of the bus cycle can result from the normal end of the bus cycle or the bus cycle can be halted.

The external transmit controller normally requests whole-bus mode during the address phase of the cycle. If whole-bus mode is not requested, then the local whole-bus mode arbiter will not grant whole-bus mode for the bus cycle. However, this rule changes during isochronous-priority mode (see below). In some embodiments, the request for whole-bus mode can be an explicit bus message. In other embodiments, the request is implicit. For example, an implicit request can be, e.g., any transfer over a particular number of bytes, e.g., 32, which is automatically treated as a request for whole bus mode. If a local half-bus is busy when the request for whole-bus mode is received from the external transmit controller and later becomes idle while the requesting transfer is still active, the local arbiter determines if any other local requesters are present, and if not grants its local half-bus to the external transmit controller.

Referring to FIG. **8**, during asynchronous-priority mode, if whole-bus mode is requested by the external side and there are no local requesters for the bus, then whole-bus mode will be granted to the externally generated bus cycle. If, during asynchronous-priority mode, whole-bus mode is requested but there are local requesters, then whole-bus mode will not be granted; the local bus cycle will be granted instead.

Referring to FIG. **9**, if, during asynchronous-priority mode, whole-bus mode has been granted to the local side and there are more local requesters at the end of the bus cycle, then the bus remains in whole-bus mode for the local requesters' bus cycles. However, if there are no more local requesters, then the link reverts back to half-bus mode at the end of the bus cycle.

Referring to FIG. **10**, if, during asynchronous-priority mode, whole-bus mode has been granted to the external side and there is a local asynchronous requester, then the local side generates a half-bus request (HalfBusReq) message and the external side is required to place the bus back into half-bus mode.

Referring to FIG. **11**, while in isochronous-priority mode, requests in the address phase of the bus cycle for whole-bus mode are ignored. If, during this mode, there are local isochronous requesters, then whole-bus mode is not granted; the local half-bus is granted to an isochronous requester instead.

If during isochronous-priority mode there are no local isochronous requesters, then whole-bus mode will be automatically granted to the other side of the link. The link will then stay in whole-bus mode, for short and long bus cycles, until the end of the last isochronous bus cycle; after that, the link will return to asynchronous-priority mode. Once in asynchronous-priority mode, the link may or may not stay in whole-bus mode, based on the rules described above.

FIG. **12** shows a traffic example including isochronous and asynchronous data and whole and half-bus mode. FIG. **12** illustrates how the arbiter optimizes for asynchronous transfers during periods of relatively loose traffic, but when asynchronous traffic dominates, the isochronous bandwidth is still guaranteed. In this example, isochronous bus cycles are assumed to consume slightly more than 50% of the available bandwidth. In the first frame **1201**, the isochronous traffic is allowed to occur early, between asynchronous bus cycles because of the lack of asynchronous traffic. At **1203**, the bus goes into isochronous whole-bus mode. After isochronous whole bus mode is over, the bus is either idle or transferring asynchronous traffic for the remainder of the frame. At the end of the frame, the bus is in whole-bus mode again except this time, transferring asynchronous data. Note that whole-bus mode may extend across frame boundaries as shown. All the isochronous requesters in frame **1201** complete their bus cycles before getting too close to the end of the frame and thus, isochronous priority mode is unnecessary.

However, in the second frame **1207**, the amount of asynchronous traffic has increased to the point where the isochronous bus cycles are in danger of not completing before the end of the frame because of large blocks of asynchronous data. Therefore, the bus enters isochronous-priority mode and the long asynchronous cycle **1209** shown in whole-bus mode is halted. After all the isochronous requesters on one side of the link have completed their bus cycles at **1211**, the bus enters into whole-bus mode for the remainder of isochronous-priority mode. Then, after the other side completes all of its remaining isochronous bus cycles in whole-bus mode, the bus is placed back into asynchronous-priority mode. After that, the asynchronous traffic is allowed to continue, through to the next frame.

Referring to FIG. **13**, a flowchart illustrates utilization of the frame counter and isochronous byte counter to enter and exit isochronous mode. At the beginning of the frame **1300** the frame counter is loaded with the number of bytes per frame and the isochronous counter is loaded at **1301** with the maximum number of isochronous bytes that can be transferred each frame. At **1302**, the frame enters asynchronous priority mode. At **1303** the frame counter is decremented at the data rate for each possible transfer across the bus (whether or not the bus is actually idle). At **1305**, it is determined if an isochronous byte was transferred. If so, the isochronous counter is decremented at **1307**. At **1309** it is determined if the elapsed-frame counter has passed the isochronous-byte counter. If so, that indicates when the remaining time in the frame is required to be allocated to the isochronous stream to guarantee that the isochronous requests can be serviced during that frame. If the frame counter is still greater than the isochronous counter, then the frame counter is decremented at **1303** on the next clock edge and the loop repeats. If however, there is a need to switch to isochronous mode, the bus switches at **1310**. At **1311** it is determined if the isochronous transfers are complete and if so, the bus returns to asynchronous priority mode at **1313**.

Thus, at **1309**, if there is at least one isochronous requester requiring the bus, then the local hardware will enter isochronous-priority mode. Any side with an isochronous requester or a current isochronous bus cycle will then send an isochronous priority mode (IsoPriMode) message to the other side, so both sides will be in isochronous-priority mode. If there is a bulk asynchronous transfer taking place, it will be halted as illustrated in FIG. **12**.

At this point, all isochronous requesters will be granted the bus, one after another, until there are no more isochronous requesters. The last isochronous bus cycle is guaranteed to complete before the end of the frame.

The first side that finishes all its isochronous bus cycles sends the asynchronous priority request (AsyPriReq) message across the bus to indicate that it is ready to enter asynchronous-priority mode. This automatically grants whole-bus mode to the other side. The second side continues isochronous bus cycles in whole-bus mode until they are complete. At this point the second side sends the asynchronous priority grant (AsyPriGnt) message, which places the link into asynchronous-priority mode.

So, to enter isochronous-priority mode, only one side needs to send the isochronous priority mode (IsoPriMode) message. However, to exit back into asynchronous-priority mode a side has to have transmitted the AsyPriReq message and the other side has to have transmitted AsyPriGnt message. If side A sends the IsoPriMode message to side B and side B has no isochronous requesters, then side B immediately responds with the AsyPriReq message. The bus then goes back into asynchronous-priority mode as soon as side A finishes its last isochronous bus cycle and sends out the AsyPriGnt message. In the rare case that side A and side B finish their last isochronous bus cycles at the same time and send out AsyPriReq messages simultaneously, then they will both send out HalfBusAck messages and the link will revert back to half-bus mode and asynchronous-priority mode.

In other embodiments, different approaches are used to ensure that isochronous guarantees are met for each frame. For instance, instead of loading the isochronous byte counter with the maximum possible isochronous byte count each frame, each link interface side could load the isochronous byte counter at the start of each frame with the actual isochronous byte count requested for the frame. That counter is then decremented for each isochronous byte transmitted. When the elapsed frame counter indicates that the time remaining in the cycle was approaching or equal to the value of the isochronous byte counter, the bus switches to the isochronous priority mode to ensure the transfer of the isochronous bytes. Such an approach would require that no more than the maximum allocable isochronous bandwidth was requested each frame. Those of skill in the art will recognize that the counters described as counting down may of course be implemented to count in other ways to determine the count value, e.g., the counters may also be implemented as up counters.

In still another approach, the bus may switch to isochronous priority mode at a predetermined time in the frame and any remaining isochronous transfers are transferred at this time. That approach may increase the latency of asynchronous data somewhat, since there may be no need to switch to isochronous priority mode.

Bus messages are transmitted over the least-significant byte of the data bus (which may vary, based on whether the bus is in whole-bus mode or half-bus mode when the byte is transmitted) while the control signal (CTLA2B, CTLB2A) is asserted. Thus, the control signals are used to send messages. When in half-bus mode or when the local side owns whole-bus mode, assertion of the local control signal indicates that the least-significant byte of the data bus is transmitting a link message. This may happen when the bus is idle or at any time during a bus cycle. The least-significant byte of the bus may vary based on whether the bus is in whole-bus mode or half-bus mode.

If whole-bus mode is granted to the external side, then the local side can assert the control signal (CTLA2B OR CTLB2A) to generate the request for half bus mode (HalfBusReq) message without the benefit of the data bus. This message informs the external side that it must relinquish whole-bus mode, even though it may be in the middle of a bulk transfer. The external side is required to accommodate this request by generating a half bus grant (HalfBusGnt) message, after which the external bus cycle will continue over the external half-bus.

A bus message can be transmitted during any CLK cycle, including when the bus is idle, during the address phase of a transfer, or during the data phase. After the message is transmitted, if there is a bus cycle taking place (and it is not requested to be stopped by the bus message), then the bus cycle will continue on the following CLK cycle, as if it were in suspended animation for a single CLK cycle. Bus messages use all the bytes that are owned by the transmitting side of the link at the time that the message is transmitted. So if the bus is in half-bus mode, the message will be placed on the least-significant byte of the local half-bus; if it is whole-bus mode then it will be placed on the least-significant byte of the whole bus.

The bus messages for one embodiment are described in Table 2

TABLE 2

Bus messages	Description
NewFrame	This is used to indicate the beginning of a new frame.
IsoPriMode	Enter isochronous-priority mode. The generation of this message, from either or both sides, places both sides into isochronous-priority rnode.
AsyPriReq	Request to enter asynchronous-priority mode and grant whole-bus mode. This message is sent by the first side to finish transmitting local isochronous bus cycles while in isochronous-priority mode. After the local side transmits the AsyPriReq message, it automatically places the local half-bus into high-impedance mode and enters whole-bus mode. In the rare case that each side both transmits and receives the AsypriReq message (simultaneously), then it automatically sends the HafBusAck message and goes back into half-bus mode (both sides will do this) and the link reverts back to asynchronous-priority mode.
AsyPriGnt	Grant asynchronous priority mode. This message is transmitted by the second side to finish transmitting local isochronous bus cycles while in isochronous-priority mode. A side determines that it is the second side to finish transmitting isochronous traffic if it has received the AsyPriReq message while in isochronous-priority mode. The AsyPriGnt message places the link into asynchronous-priority mode on the following clock.
HaltXfer	Halt the current bulk transfer in progress. The bus cycle that is in progress is immediately stopped by this message.
WholeBusGnt	Whole-bus mode grant. This is sent from the side granting whole-bus mode to the side requesting whole bus mode. After it is sent by the local side, the local side places its half-bus into high-impedance mode.
WholeBusAck	Whole-bus mode acknowledge. This is sent from the side requesting whole-bus mode, after the other side has sent either the WholeBusGnt or the AsyPriReq message, to indicate that data will be transferred over the whole bus starting on the following clock. The link is still considered to be in half-bus mode while this message is being transmitted.
HalfBusReq	When the external side has been granted whole-bus mode, the local side requests that the link be placed back into half-bus mode by asserting the CTL pin for one CLK cycle. Since the local side controls no data bus pins, there is no decode for this message.
HalfBusGnt	The owner of whole-bus mode sends this message in response to the HalfBusReq message to indicate that it is reverting back into half-bus mode. On the clock following this message, the link is considered to be in half-bus mode.

TABLE 2-continued

Bus messages	Description
HalfBusAck	When the external side owns whole-bus mode and it is ended (by a HalfBusGnt, HaltXfer, or an EndXfer), then this message is sent by the local side to indicate that it is back in control of the half-bus.

In addition to the message above, bus messages may be used for legacy signal transfers. Legacy signals are those signals in personal computer (or other) systems that are necessitated by the desire to ensure compatibility with older designs. Thus, the interface module may need to transmit legacy signal information to the processor module. In one embodiment bits(3:0) of the message are used to transmit the legacy signals STPCLK#, IGNNE#, CPURST, A20M#. Those legacy signals are known in the art and are not discussed further herein. The interface module (IM) transmits this data each time one of those signals change state.

Further, bus messages may be used to transfer interrupt signals states from the interface module to the processor module. For this bus message bits(3:0) are used to transfer interrupts SMI, NMI, INTR, INIT. The interface module transmits this data each time one of these signals change state.

Finally, legacy information may be needed to be transferred from the processor module to the interface module. For example, FERR# may be transferred in a specific bit position of a bus message. The processor module transmits this data each time one of the signals changes state.

If transferring of legacy information conflicts unduly with whole-bus mode and/or isochronous-priority mode operations, separate signals may be utilized in the computer system for transferring some or all of the legacy and interrupt signals between the processor module and the interface module.

As discussed above, certain situations result in the halt of a bus cycle. For instance, while in asynchronous-priority mode, if a bulk isochronous transfer is taking place and there is an asynchronous requester, then the transfer will be halted to minimize the latency of the asynchronous bus cycle. After entering isochronous-priority mode, if a bulk asynchronous transfer is taking place, it will be halted. The bus is then only granted to isochronous requesters during isochronous-priority mode.

Bus cycles are not halted during their address phase, only during their data phase. The requester of a halted bus cycle continues to request the cycle. Once it is granted again, data continues from the point where it left off.

Now that the interconnect bus is understood, various computer system architectures exploiting features of the bus may be further examined. Another architectural variant of FIG. 2 is shown in FIG. 14. In this embodiment, graphics subsystem 214 has been moved from north bridge 210 to south bridge 211. That significantly reduces the pin count of the north bridge integrated circuit. Reduced pin count generally results in lower cost. That also allows data to be routed to the graphics subsystem 214 directly from 1394 host controller 219. Thus, e.g., data from a video camera (not shown) could be routed directly through the HVLPC interface controller 213 through AGP bridge 1405 to graphics subsystem 214.

In still another option (not shown), the graphics controller 1401 of graphics subsystem 214 may be located in south bridge 211.

With the graphics bus and/or the graphics controller integrated into south bridge 211, certain graphical functions may remain in the north bridge 201. When the graphics port being used is the accelerated graphics port (AGP), the AGP lets the video processor in video/graphics controller 1401 access system memory 207 for graphics calculations. Graphics controllers typically access contiguous data structures in their local memory (frame buffer 1403) but if the data structures are stored in system memory, the structures can be dynamically allocated. Therefore the graphics controller needs to remap local memory addresses to system memory. The Graphics Address Remapping Table (GART) remaps addresses from the local memory for the graphics controller to the system memory. It is more convenient if that function is maintained close to the memory and therefore still resides in north bridge 201 in the embodiment described in FIG. 14.

Referring to FIG. 15 another implementation exploiting the interconnection bus is shown in which the north bridge and the CPU are integrated together into a single integrated circuit 1501. That integration significantly reduces the pin count of the processor integrated circuit that incorporates the CPU. The reduction in pin count results from the removal of the bus interface, e.g. the Socket 7 bus which is shown in FIG. 14 as the host bus. A three port switch 1503 in CPU integrated circuit 1501 connects the memory controller 1505, CPU 1507, and the HV-LPC bus interface controller 1509. Note that the memory controller may include a Rambus controller to interface to system memory using RDRAM.

Assuming the memory interface is a low pin count interface such as Rambus or Sinclink, and given the low pin count of the HV-LPC interconnection bus, the total pin count of the system is reduced which results in a reduction in cost. For example, one architecture in use today uses a 321-pin CPU and a 328-pin north bridge chip. Where the north bridge is integrated with the CPU, the entire north bridge/CPU function can be reduced to a single integrated circuit with less than 328 total pins. Thus, the embodiment of FIG. 15 has eliminated an entire integrated circuit. In addition, the reduction in pins results in further savings caused by saving board space and routing.

In FIG. 15, the AGP bridge 1512 is coupled to the link controller 1514. The graphics controller can communicate with system memory through the interconnection bus. Additionally, the 1394 bus can communicate directly with AGP bridge 1512 through the link controller without going through system memory (RDRRAM first). That reduces traffic on the bus and ensures isochronous data is provided to its ultimate destination more quickly than if the isochronous data had to be stored first in system memory 207 using memory controller 1505.

Referring to FIG. 16, another embodiment is shown in which the video/graphics controller 1401 is integrated with the South Bridge integrated circuit and connected to link controller 1514. As data intensive applications grow, the throughput and latency demands on the interconnect bus will also grow. Therefore, it is important that the interconnect bus described herein supports high bandwidth asynchronous and isochronous data for both present and future applications. Note that placing the graphics controller in the south bridge or attached to the south bridge and using the same interconnect bus for the graphics data as well as other input/output system data, can place heavy throughput and latency requirements on the interconnect bus.

Referring to FIG. 17, another embodiment is shown in which the video/graphics controller 1401 is integrated with

the CPU, memory controller and the interconnect bus controller in integrated circuit **1701**. Integrating the graphics controller on the integrated circuit that includes the processor core provides the advantage of greater system integration resulting in the elimination of a separate graphics controller circuit.

Further pin savings can be effected by utilizing a unified memory architecture (UMA) in which the system RAM (e.g. RDRAM **207**) is used in place of frame buffer **1403**. The frame buffer **1403** is a memory holding the video image and is typically found on graphic boards. Typically, computer systems have provided the graphics controller on a separate graphics card. Utilizing UMA results in a lower pin count because the interface from CPU integrated circuit **1701** to frame buffer **1403** is eliminated. However, that comes at the expense of system performance because all of the graphics controller to frame buffer data traffic is handled by system memory in the UMA approach.

Note that in some embodiments, it may be necessary to provide sideband signal(s) directly from the south bridge integrated circuit to the processor when the latency guaranteed on the HV-LPC is too long for certain functions that have shorter requirements. For example, if power management in the south bridge knows that the processor is going to have its clocks shut down in 100 nanoseconds, the latency inherent with the interconnection bus may too long to inform the processor prior to the event, thus necessitating the sideband signal(s).

The description of the invention set forth herein is illustrative, and is not intended to limit the scope of the invention as set forth in the following claims. Variations and modifications of the embodiments disclosed herein, may be made based on the description set forth herein, without departing from the scope and spirit of the invention as set forth in the following claims.

What is claimed is:

1. A computer system comprising:

a processor integrated circuit including a central processing unit;

a first integrated circuit coupled by a host bus to the processor integrated circuit;

a second integrated circuit; and

an interconnection bus coupling the first and second integrated circuits, the interconnection bus transferring data in frames, the interconnection bus coupled to transfer data in asynchronous priority mode during a first part of each frame, in which asynchronous traffic is transferred instead of isochronous traffic if there are any asynchronous requesters for the interconnection bus and to transfer data in isochronous priority mode during other parts of at least some frames.

2. The computer system as recited in claim **1** wherein the second integrated circuit includes a bridge circuit coupling the interconnection bus to a peripheral component interface (PCI) bus.

3. The computer system as recited in claim **2** wherein the second integrated circuit includes a serial interface circuit coupling a serial bus transferring at least some isochronous data, to the second interconnection bus interface circuit.

4. The computer system as recited in claim **3** wherein the serial bus is an IEEE **1394** compatible bus.

5. The computer system as recited in claim **1** wherein the interconnection bus transfers data simultaneously in two directions between the first and second integrated circuits.

6. The computer system as recited in claim **5** further comprising a switch circuit coupled to the host bus, for selectably coupling the system memory, the processor integrated circuit, and the interconnection bus.

7. The computer system as recited in claim **1** further comprising system memory coupled to the first integrated circuit, the system memory including random access memory (RAM) devices and wherein the first integrated circuit includes a memory controller for the system memory.

8. The computer system as recited in claim **7** further comprising:

a graphics controller circuit coupled to the first bridge integrated circuit via a graphics bus.

9. The computer system as recited in claim **8** further comprising a switch circuit coupled to the graphics bus, the host bus, the first interconnection bus interface circuit, and the system memory, the switch circuit for selectably coupling the system memory to one of the processor integrated circuit, the interconnection bus and the graphics bus, the switch also selectably coupling the processor integrated circuit to the interconnection bus.

10. The computer system as recited in claim **1** further comprising:

a graphics controller circuit; and

a graphics bus connecting the graphics controller circuit to the second bridge interface circuit.

11. The computer system as recited in claim **1** further comprising a graphics controller circuit integrated on the second bridge integrated circuit.

12. A method of communicating information in a computer system that includes a processor coupled to a first bridge integrated circuit by a host bus and an interconnection bus connecting a second bridge integrated circuit to the first bridge integrated circuit, the method comprising:

transferring information between the first and second bridge integrated circuits in asynchronous priority mode in which asynchronous information is transferred instead of isochronous information if there are any asynchronous requesters, during a first part of each frame;

selectably transferring information between the first and second bridge integrated circuits in isochronous priority mode, in which transfer of isochronous information is given priority over transfers of asynchronous information.

13. The method as recited in claim **12** further comprising:

sending a message regarding impending power states over the interconnection bus.

14. The method as recited in claim **12** further comprising sending interrupt requests to the processor on the interconnection bus.

15. The method as recited in claim **12** further comprising communicating impending clock stoppages via messages sent over the interconnection bus.

16. The method as recited in claim **12** further comprising:

transferring data between the first bridge integrated circuit and a **1394** bus via the interconnection bus using isochronous priority mode.

17. The method as recited in claim **12** further comprising

selectably coupling via a switching circuit, a main memory coupled to the first bridge integrated circuit, to one of the processor and the interconnect bus; and

selectably coupling the processor via the switching circuit to one of the main memory and the interconnect bus.

18. The method as recited in claim **17** further comprising selectably coupling a graphics controller via the switching circuit to the main memory.

19. The method as recited in claim **12** further comprising transferring isochronous data from a serial bus coupled to

the second integrated circuit to the graphics controller, without storing the isochronous data in system memory.

20. The method as recited in claim **12** further comprising:

transferring data between the first bridge integrated circuit and devices resident on a peripheral component interface (PCI) bus via the interconnection bus in the asynchronous priority mode; and

transferring data between the first bridge integrated circuit and a **1394** bus via the interconnection bus using the isochronous priority mode.

21. An integrated circuit comprising:

a plurality of input and output terminals for connecting to an interconnection bus;

an interface coupled to the input and output terminals, the interface coupled to send and receive data in frames, the interface responsive to transfer information in asynchronous priority mode during a first part of a frame and in isochronous priority mode during other parts of the frame, the interconnect bus selectably switching to isochronous priority mode during the frame according to an amount of isochronous traffic transferred during the frame.

22. The integrated circuit as recited in claim **21** wherein the integrated circuit includes a memory controller and is coupled to a microprocessor via a host bus.

23. The integrated circuit as recited in claim **21** wherein the integrated circuit includes a bridge circuit to couple the interconnection bus to a peripheral component interconnect (PCI) bus.

24. The integrated circuit as recited in claim **21** wherein the interconnect bus switches to isochronous priority mode according to the amount of isochronous traffic transferred during asynchronous priority mode.

* * * * *

US006016528A

United States Patent [19]

Jaramillo et al.

[11] Patent Number: 6,016,528

[45] Date of Patent: Jan. 18, 2000

[54] **PRIORITY ARBITRATION SYSTEM PROVIDING LOW LATENCY AND GUARANTEED ACCESS FOR DEVICES**

[75] Inventors: **Ken Jaramillo; David Gerard Spaniol**, both of Phoenix, Ariz.

[73] Assignee: **VISI Technology, Inc.**, San Jose, Calif.

[21] Appl. No.: **08/960,184**

[22] Filed: **Oct. 29, 1997**

[51] **Int. Cl.**[7] **G06F 13/14**

[52] **U.S. Cl.** **710/243**; 710/244; 710/107; 710/240

[58] **Field of Search** 395/732, 800, 395/800.34, 325, 725; 710/107, 123, 241, 242, 243, 244, 240

[56] **References Cited**

U.S. PATENT DOCUMENTS

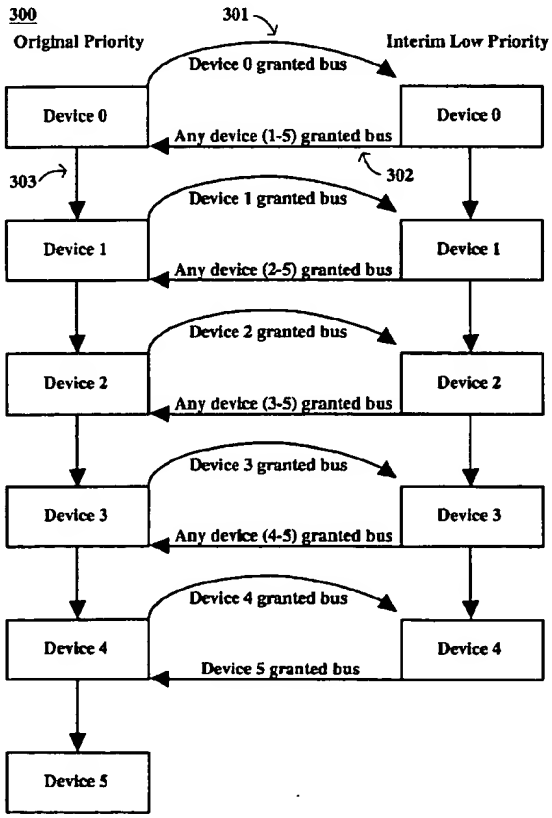
5,392,436	2/1995	Jansen et al.	395/725
5,396,602	3/1995	Amini et al.	395/325
5,619,720	4/1997	Garde et al.	395/800
5,805,905	9/1998	Biswas et al.	395/732
5,822,768	10/1998	Shakkarwar	711/149
5,826,101	10/1998	Beck et al.	395/800.34

Primary Examiner—Ayaz R. Sheikh
Assistant Examiner—Tim Vo
Attorney, Agent, or Firm—Wagner, Murabito & Hao LLP

[57] **ABSTRACT**

The present invention comprises a priority arbitration system for interfacing a plurality of PCI agents coupled to a peripheral component interconnect (PCI) bus such that high priority PCI agents are satisfied without starving low priority PCI agents. The system of the present includes a PCI bus adapted to transmit data signals. At least one high priority PCI agent is coupled to the PCI bus. At least one low priority PCI agent is coupled to the PCI bus. An arbiter is coupled to the high priority PCI agent and the low priority PCI agent via the PCI bus. The arbiter grants ownership of the PCI bus to the high priority PCI agent prior to granting ownership to the low priority PCI agent. After being granted ownership, the high priority PCI agent becomes an interim low priority PCI agent. The low priority PCI agent is accorded a higher priority by the arbiter than the interim low priority PCI agent. The interim low priority PCI agent reverts to the high priority PCI agent subsequent to a grant to the low priority PCI agent. In this manner, the arbiter, by granting ownership of the PCI bus to the low priority PCI agent before granting ownership of the PCI bus to the interim low priority PCI agent, ensures the low priority PCI agent is not prevented from accessing the PCI bus by the high priority PCI agent.

18 Claims, 5 Drawing Sheets



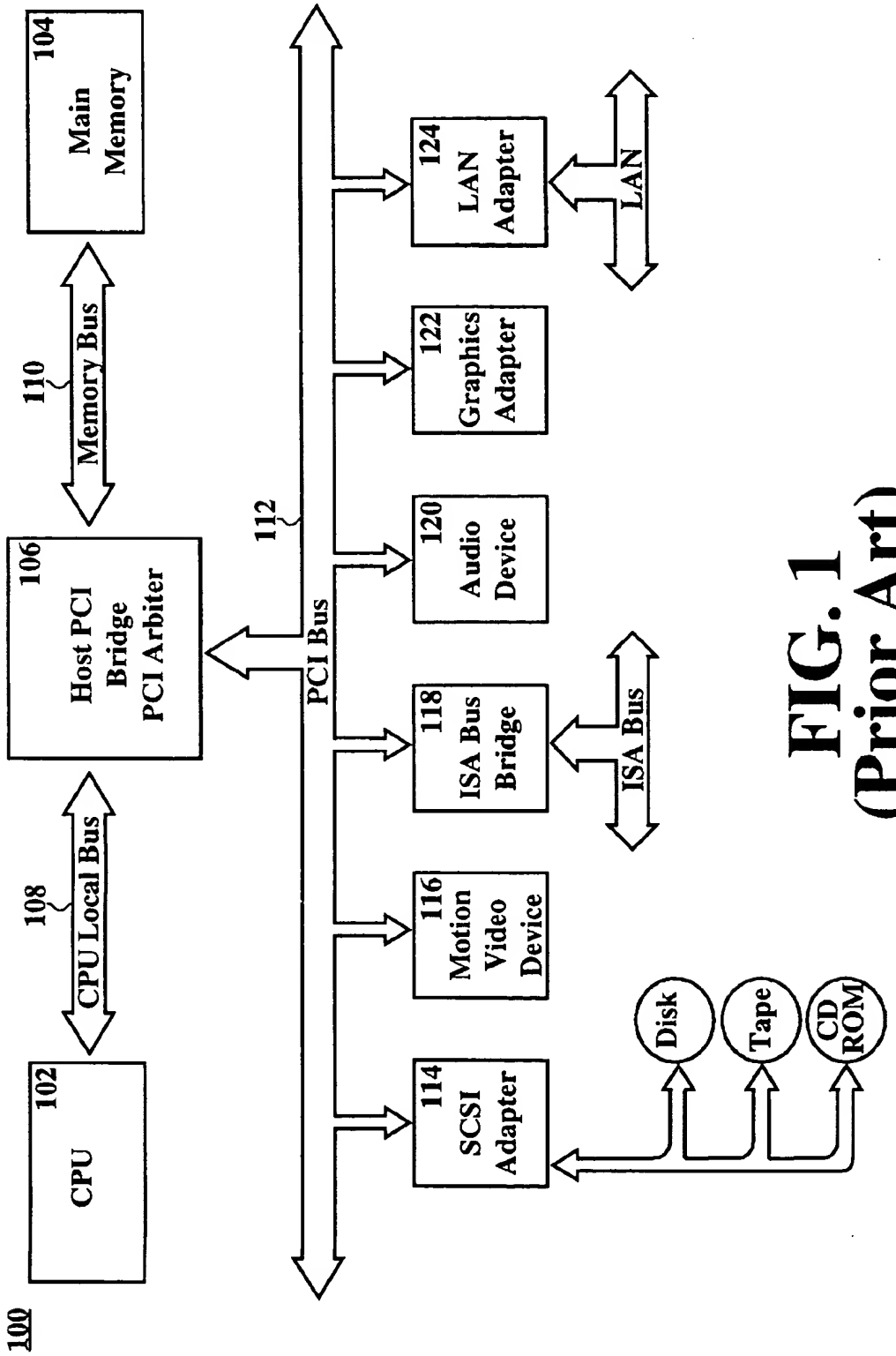


FIG. 1
(Prior Art)

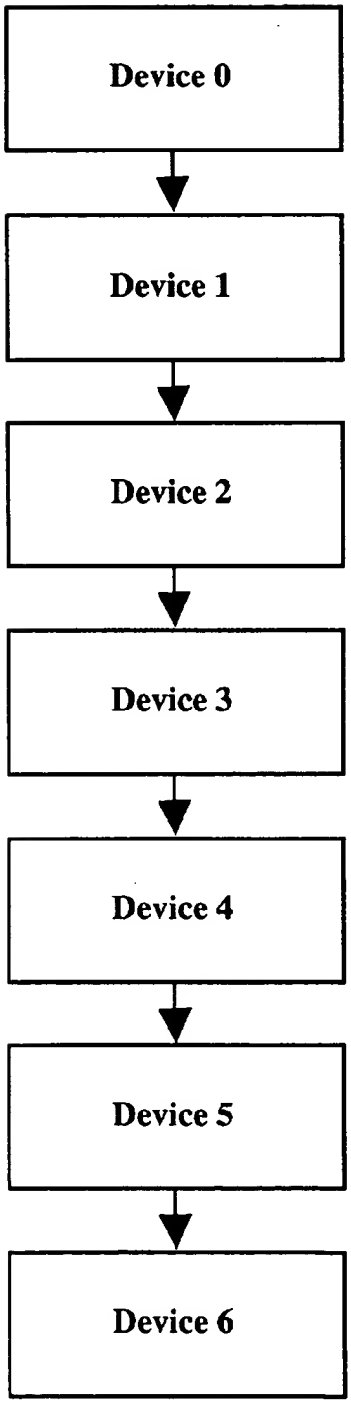


FIG. 2
(Prior Art)

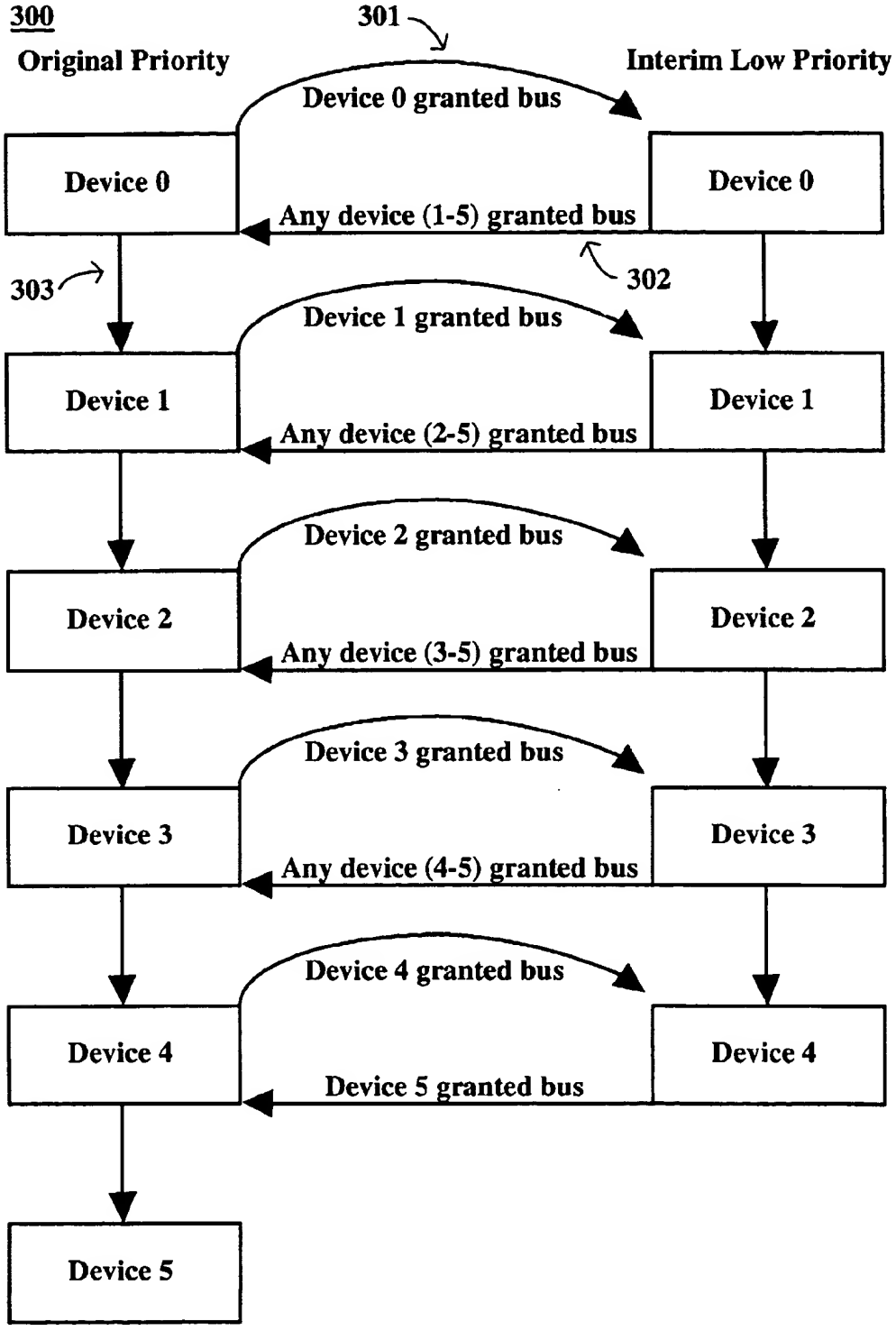

300
301
Original Priority
Interim Low Priority
Device 0 granted bus
Device 0
Device 0
Any device (1-5) granted bus
303
302
Device 1 granted bus
Device 1
Device 1
Any device (2-5) granted bus
Device 2 granted bus
Device 2
Device 2
Any device (3-5) granted bus
Device 3 granted bus
Device 3
Device 3
Any device (4-5) granted bus
Device 4 granted bus
Device 4
Device 4
Device 5 granted bus
Device 5

FIG. 3

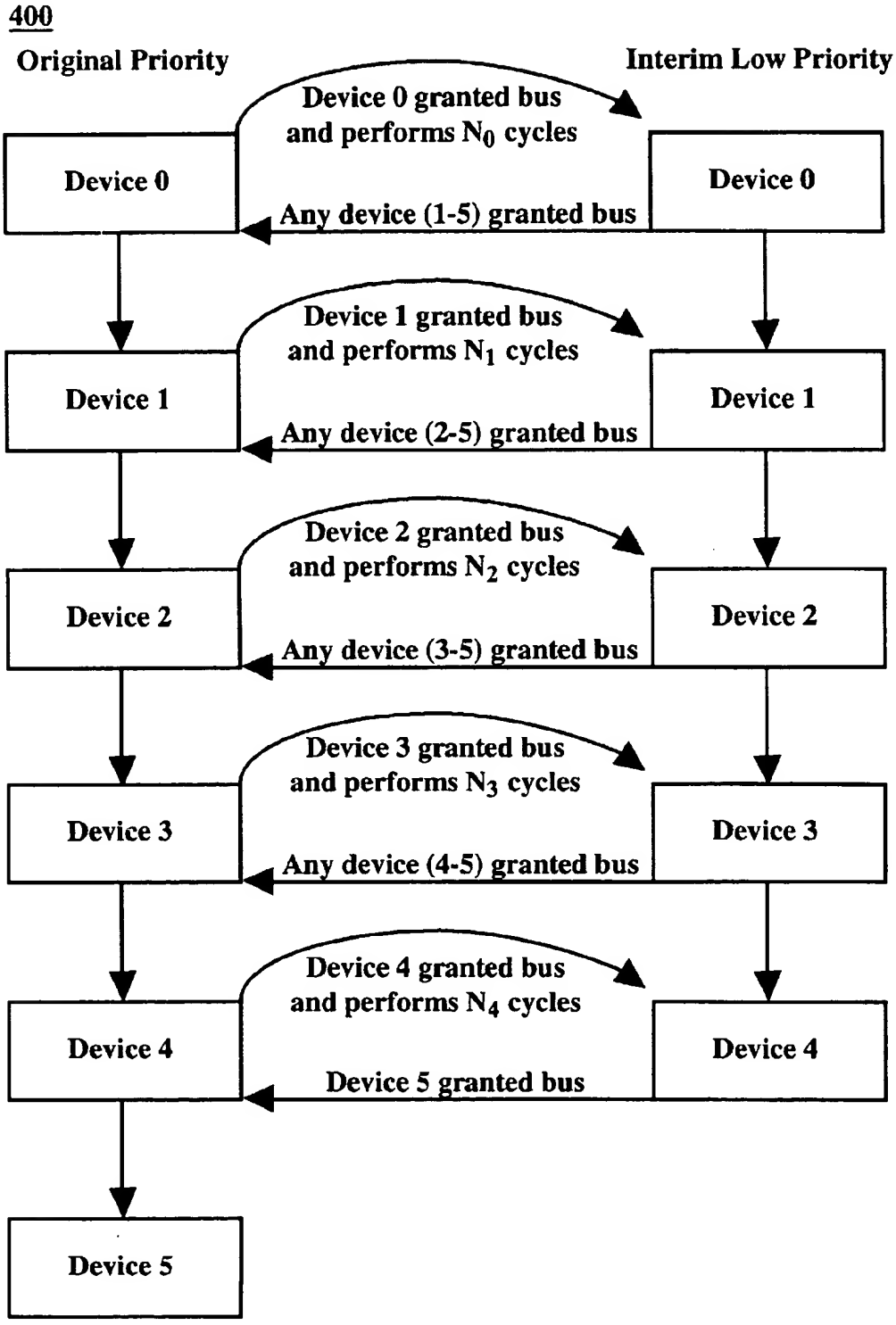

400
Original Priority
Interim Low Priority
Device 0 granted bus and performs N_0 cycles
Any device (1-5) granted bus
Device 0
Device 0
Device 1 granted bus and performs N_1 cycles
Any device (2-5) granted bus
Device 1
Device 1
Device 2 granted bus and performs N_2 cycles
Any device (3-5) granted bus
Device 2
Device 2
Device 3 granted bus and performs N_3 cycles
Any device (4-5) granted bus
Device 3
Device 3
Device 4 granted bus and performs N_4 cycles
Device 5 granted bus
Device 4
Device 4
Device 5

FIG. 4

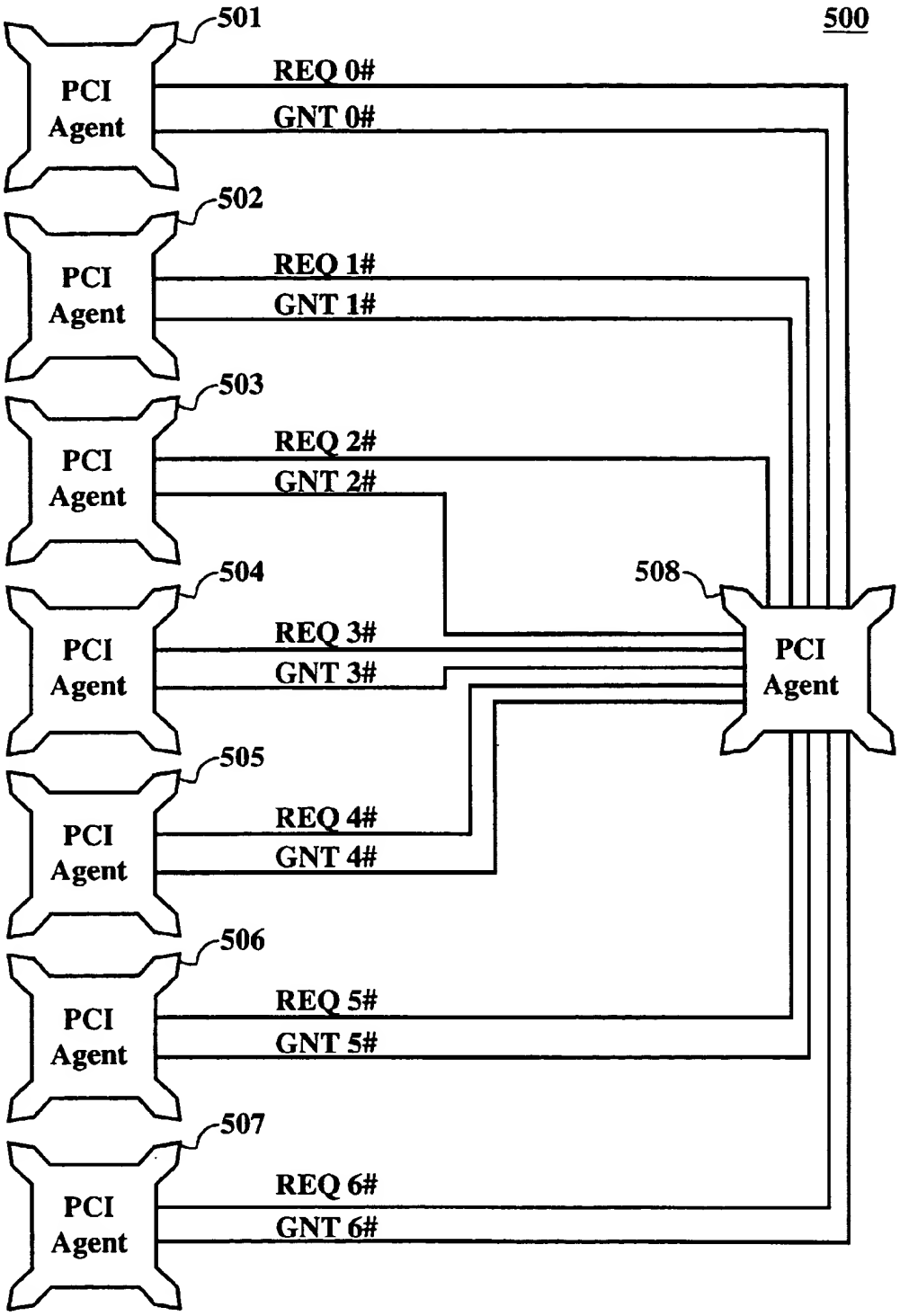

500
501
PCI Agent
REQ 0#
GNT 0#
502
PCI Agent
REQ 1#
GNT 1#
503
PCI Agent
REQ 2#
GNT 2#
504
PCI Agent
REQ 3#
GNT 3#
505
PCI Agent
REQ 4#
GNT 4#
506
PCI Agent
REQ 5#
GNT 5#
507
PCI Agent
REQ 6#
GNT 6#
508
PCI Agent

FIG. 5

PRIORITY ARBITRATION SYSTEM PROVIDING LOW LATENCY AND GUARANTEED ACCESS FOR DEVICES

TECHNICAL FIELD

The present invention pertains to the field of arbitration systems for computer system bus architectures. More particularly, the present invention relates to fixed priority arbitration system for providing low bus latency for critical devices and guaranteed access for non-critical devices.

BACKGROUND ART

A bus architecture of a computer system conveys much of the information and signals involved in the computer system's operation. In a typical computer system, one or more busses are used to connect a central processing unit (CPU) to a memory and to input/output elements so that data and control signals can be readily transmitted between these different components. When the computer system executes its programming, it is imperative that data and information flow as fast as possible in order to make the computer as responsive as possible to the user and to efficiently execute its assigned tasks. With many peripheral devices, such as graphics adapters, full motion video adapters, small computer systems interface (SCSI) host bus adapters, and the like, it is imperative that large block data transfers be accomplished expeditiously for efficient execution. These devices are just some examples of subsystems which benefit substantially from a very fast bus transfer rate.

Much of a computer system's functionality and usefulness to a user is derived from the functionality of the peripheral devices. For example, the speed and responsiveness of the graphics adapter is a major factor in a computer system's usefulness as an entertainment device. Or, for example, the speed with which video files can be retrieved from a hard drive and played by the graphics adapter determines the computer system's usefulness as a training aid. Hence, the rate at which data can be transferred among the various peripheral devices often determines whether the computer system is suited for a particular purpose. The electronics industry has, over time, developed several types of bus architectures. Recently, the PCI (peripheral component interconnect) bus architecture has become one of the most widely used, widely supported bus architectures. The PCI bus was developed to provide a high speed, low latency bus architecture from which a large variety of systems could be developed.

Prior Art FIG. 1 shows a typical PCI bus architecture **100**. PCI bus architecture **100** is comprised of a CPU **102** and a main memory **104**, coupled to a host PCI bridge containing arbiter **106** (hereafter arbiter **106**) through a CPU local bus **108** and memory bus **110**, respectively. A PCI bus **112** is coupled to each of PCI agents **114**, **116**, **118**, **120**, **122**, **124** respectively, and is coupled to arbiter **106**.

Referring still to Prior Art FIG. **1**, each of PCI agents **114**, **116**, **118**, **120**, **122**, **124** (hereafter, PCI agents **114–124**) residing on PCI bus **112** use PCI bus **112** to transmit and receive data. PCI bus **112** is comprised of functional signal lines, for example, interface control lines, address/data lines, error signal lines, and the like. Each of PCI agents **114–124** are coupled to the functional signal lines comprising PCI bus **112**. When one of PCI agents **114–124** requires the use of PCI bus **112** to execute a data transaction, it requests PCI bus ownership from arbiter **106**. The PCI agent requesting ownership is referred to as an "initiator", or bus master. Upon being granted ownership of PCI bus **112** from arbiter **106**, the initiator (e.g., PCI agent **116**) carries out its respective data transaction (e.g., transfer a file).

Each of PCI agents **114–124** may independently request PCI bus ownership. Thus, at any given time, several of PCI agents **114–124** may be requesting PCI bus ownership simultaneously. Where there are simultaneous requests for PCI bus ownership, arbiter **106** arbitrates between requesting PCI agents, determining which requesting PCI agent is granted PCI bus ownership. When one of PCI agents **114–124** is granted PCI bus ownership, it initiates its transaction (e.g., file transfer) with a "target" or slave device (e.g., main memory **104**). When the data transaction is complete, the PCI agent relinquishes ownership of PCI bus **112**, allowing arbiter **106** to reassign PCI bus **112** to another requesting PCI agent.

Thus, only one data transaction can take place on a PCI bus at any given time. In order to maximize the efficiency and data transfer bandwidth of PCI bus **112**, PCI agents **114–124** follow a definitive set of protocols. These protocols are designed to standardize the method of accessing, utilizing, and relinquishing PCI bus **112**, so as to maximize its data transfer bandwidth while ensuring interoperability among various PCI bus devices from various manufacturers. The PCI bus protocols and specifications are set forth in an industry standard PCI specification (e.g., PCI Specification—Revision 2.1). Where each of PCI agents **114–124** are high performance, well designed devices, data transfer rates approaching 528 Mbytes per second can be achieved (e.g., a 64 bit PCI bus **112** operating at 66 MHz).

Sustaining consistently high data transfer rates across a PCI bus requires efficient allocation of PCI bus ownership, and thus, PCI bus bandwidth. Each device needs ownership of the PCI bus in accordance with its respective requirements. These requirements include, for example, latency tolerance, data transfer bandwidth, block transfer size, and the like. Some devices are more critical to the proper operation of the computer system than others. Some devices are less tolerant of latency than others. In addition, some devices need to transfer very large quantities of data. Hence, devices coupled to the PCI bus are of differing priority with regard to their respective requests for PCI bus ownership.

Efficiently managing the allocation of the PCI bus is essential to the proper operation of a computer system. Efficient allocation management assumes even greater importance when devices of differing priority require ownership of the PCI bus. As described above, only one device at a time can transfer data across the PCI bus. Consequently, competing devices arbitrate for ownership and the arbiter determines which device is granted the PCI bus. Afterwards, the remaining devices subsequently are forced to wait and continue arbitration. Some devices are more important to the functionality of the computer system than other devices and are thus considered high priority. Some devices are more tolerant of latency than other devices and are accordingly considered lower priority. The arbiter needs to ensure the bus is allocated among the competing devices, taking into account their relative priorities. To accomplish this, the arbiter follows a predetermined arbitration methodology, or arbitration scheme.

Prior Art FIG. **2** shows a typical prior art prioritized arbitration scheme **200**. Arbitration scheme **200** shows the relative priority of 7 coupled devices, device **0** through device **6**, where device **0** is the highest priority device and device **6** is the lowest priority device. Higher priority devices are allocated the higher priority positions in arbitration scheme **200** (e.g., device **0**) while lower priority

devices are allocated lower priority positions (e.g., device 6). For example, a network adapter card typically is required to transfer very large blocks of data from the network to main memory, which requires a disproportionately large amount of PCI bus data transfer bandwidth. The network adapter also typically has internal buffers of limited size, which cannot tolerate data transfer latency without incurring a buffer overrun or underrun. Consequently, the network adapter would be coupled as device 0. In contrast, a printer does not transfer particularly large blocks of data, is very tolerant of latency, and is thus coupled as device 6. In this manner, peripheral devices are coupled to the PCI bus and are granted ownership of the PCI bus according to their respective priorities. Hence, where all of devices 0 through 6 simultaneously request ownership, device 0 receives ownership first.

There is a problem, however, in that arbitration scheme 200 does not adequately match the differing requirements of bandwidth and priority of different devices. In arbitration scheme 200, where there are many high bandwidth devices using the PCI bus and where each issues successive requests for ownership, lower priority devices can be prevented from acquiring ownership for long periods of time, or "starved" of PCI bus bandwidth. Arbitration scheme 200 does not ensure low priority devices are not prevented from acquiring ownership.

For example, at any time device 6 and any other device request ownership, device 6 "loses" the arbitration and is required to wait for a successive arbitration following the resulting data transaction. Where there are numerous requests for access from other devices, each one of which is of a higher priority than device 6, there may be very prolonged periods of time during which device 6 never aquires ownership. To ensure proper operation of the computer system, higher priority devices, which typically require large amounts of data transfer bandwidth, occupy the higher priority positions. While this assists the higher priority devices in receiving their required data transfer bandwidth, low priority devices can be easily starved.

Thus, what is required is an arbitration scheme which is much more flexible with regard to allocating PCI bus bandwidth. The required solution needs to ensure low priority devices are not starved while ensuring high priority devices are adequately served. The required system needs to efficiently allocate PCI bus bandwidth to maximize the overall functionality of a computer system. The present invention provides a novel solution to the above requirements.

DISCLOSURE OF THE INVENTION

The present invention provides a method and system of priority arbitration which provides predictable latency and guaranteed access for devices coupled to a PCI bus. The present invention provides an arbitration scheme which is much more flexible with regard to allocating PCI bus bandwidth. The present invention ensures low priority devices are not starved while ensuring high priority devices are adequately served. The present invention efficiently allocates PCI bus bandwidth to maximize the overall functionality of a computer system.

In one embodiment, the present invention comprises a priority arbitration system for interfacing a plurality of PCI agents coupled to PCI bus such that high priority PCI agents are satisfied without starving low priority PCI agents. The system of the present includes a PCI bus adapted to transmit data signals, at least one high priority PCI agent coupled to the PCI bus, and at least one low priority PCI agent coupled to the PCI bus. An arbiter is coupled to the high priority PCI agent and the low priority PCI agent via the PCI bus. In accordance with their respective priority, the arbiter grants ownership of the PCI bus to the high priority PCI agent prior to granting ownership to the low priority PCI agent. However, in the present invention, after being granted ownership, the high priority PCI agent becomes an interim low priority PCI agent.

The low priority PCI agent is accorded a higher priority by the arbiter than the interim low priority PCI agent. Hence, the low priority PCI agent wins any arbitration against the interim low priority PCI agent. This ensures the low priority agent is not starved of bus access. Subsequent to a data transaction by the low priority PCI agent, the interim low priority PCI agent reverts back to the high priority PCI agent, which, once again, wins arbitration against the low priority PCI agent. In this manner, the arbiter, by granting ownership of the PCI bus to the low priority PCI agent before granting ownership of the PCI bus to the interim low priority PCI agent, ensures the low priority PCI agent is not prevented from accessing the PCI bus by the high priority PCI agent. Thus, the present invention efficiently allocates PCI bus bandwidth to maximize the overall functionality of the computer system.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention:

Prior Art FIG. 1 shows a typical PCI bus architecture of the prior art.

Prior Art FIG. 2 shows a diagram of a typical prior art prioritized arbitration scheme.

FIG. 3 shows a diagram of a first arbitration process in accordance with one embodiment of the present invention.

FIG. 4 shows a diagram of a second arbitration process in accordance with a more versatile alternative embodiment of the present invention.

FIG. 5 shows a diagram of an arbitration system in accordance with one embodiment of the present invention.

BEST MODE FOR CARRYING OUT THE INVENTION

Reference will now be made in detail to the preferred embodiments of the invention, a priority arbitration system providing a predictable latency and guaranteed access for devices, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with the preferred embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one of ordinary skill in the art that the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

The present invention provides a method and system of priority arbitration which provides predictable latency and

guaranteed access for devices coupled to a bus. The present invention provides an arbitration process which is much more flexible with regard to allocating bus bandwidth. The present invention ensures low priority devices are not starved while ensuring high priority devices are adequately served. In so doing, the present invention efficiently allocates bus bandwidth to maximize the overall functionality of a computer system. The present invention and its benefits are further described below.

Referring to FIG. **3**, a diagram of a priority arbitration process **300** in accordance with one embodiment of the present invention is shown. Arbitration process **300** shows the relative priority of 6 coupled devices, device **0** through device **5**, arranged in order of priority where device **0** is the highest priority device and device **5** is the lowest priority device. Higher priority devices are allocated higher priority positions in arbitration process **300** (e.g., device **0**) and lower priority devices are allocated lower priority positions (e.g., device **5**). Arbitration process **300** also shows devices **0–4** in their original priority states (e.g., on the left side) and devices **0–4** in their interim low priority states (e.g., on the right side).

High priority devices are those devices which require large amounts of data transfer bandwidth or are intolerant of latency. Such devices typically include, for example, network adapter cards, graphics accelerator cards, real time video devices, and the like. Low priority devices include, for example, telephone modems, audio cards, printers, and the like, which can function nominally under moderate latency. These devices are coupled to a PCI bus including the system of the present invention such that the highest priority device is coupled as device **0**, the next highest priority device is coupled as device **1**, and so on, through the lowest priority device being coupled as device **5**.

Arbitration process **300** functions by efficiently allocating PCI bus bandwidth (e.g., PCI bus ownership) among devices **0–5** in accordance with their respective priorities. Arbitration process **300**, in accordance with the present invention, satisfies the bandwidth and latency requirements of the high priority devices while concurrently ensuring the lower priority devices are not starved. Each of devices **0–5** are assigned a fixed, original priority according to their respective requirements. However, even though an original priority is assigned, the priority is alterable such that successive grants of the PCI bus affect future grants. In other terms, the grants of the PCI bus in a system in accordance with arbitration process **300** are "perceptible" in that successive grants to a high priority device affect subsequent grants to that high priority device.

Subsequent grants are affected through the use of interim priority states. Where a high priority PCI agent (e.g., device **0**) requests and is granted the PCI bus, it becomes an interim low priority PCI agent for the next arbitration subsequent to its data transaction. This interim low priority is treated by an arbiter in accordance with the present invention as the lowest priority of the coupled devices (e.g., devices **0–5**). The interim low priority PCI agent remains the lowest priority device until a device (e.g., device **1**) with a lower original priority is granted the bus. The interim low priority PCI agent then reverts to its state as a high priority PCI agent.

For example, device **0**, since it occupies a high priority position in arbitration process **300**, will "win" any arbitration with another device (e.g., device **1**). In accordance with the present invention, upon being granted the PCI bus, device **0** becomes an interim low priority PCI agent. This is shown by arrow **301**. At this time, device **0** has the lowest "effective" priority of devices **0–5**. This is shown by device **0** being in the interim priority column as a result of arrow **301**. For any subsequent arbitration, any of devices **1-5** (e.g., in the original priority column) will win ownership as against device **0** by virtue of device **0**'s interim low priority status. In the present embodiment, while in the interim low priority state, device **0** cannot be granted ownership unless there are no other original priority devices requesting ownership. Once one of devices **1–5** is granted ownership, device **0** reverts back to its original priority (e.g., the original priority column). This is shown by arrow **302**. Upon reversion to its original priority, device **0** once again wins any arbitration with any of devices **1–5**. This is shown by arrow **303**.

Thus, in accordance with the present invention, when any of devices **1–4** is granted the PCI bus, that device is given an interim low priority having the lowest effective priority as against all other original priority devices. It cannot be granted the PCI bus unless there are no other lower priority devices (according to the original priority column) requesting ownership. If two devices are in the interim low priority state they are assigned a priority the reverse of their original priority. For example, if device **2** and device **3** are in their interim low priority state, they have a lower effective priority than all devices in the original priority state, however, the effective priority of device **3** is still higher than that of device **2**. It should be appreciated that in the present embodiment, the device having the lowest original priority (e.g., device **5**) remains in its original priority state since it already looses any arbitration with any of devices **1–4**. However, this device is guaranteed access to the PCI bus due to the fact that as each of devices **1–4** are granted ownership, they become interim priority devices, such that, eventually, the device having the lowest original priority (e.g., device **5**) will win an arbitration for ownership.

In this manner, arbitration process **300** ensures the lower priority devices are not starved of data transfer bandwidth. The interim low priority states of arbitration process **300** avoid the PCI bus monopolization problems encountered in prior art PCI bus arbitration process. Arbitration process **300** has no potential PCI agent starvation conditions due to the fact that, in accordance with the present invention, it gives each device a chance to obtain PCI bus ownership while simultaneously preventing high priority devices from monopolizing ownership.

In addition, arbitration process **300** offers the best latency performance for the most latency sensitive devices (e.g., device **0**). For example, under worst case conditions (e.g., all devices simultaneously requesting PCI bus ownership) arbitration process **300** grants ownership to device **0** ½ of the time, device **1** ¼ of the time, device **2** ⅛ of the time, device **3** 1/16 of the time, device **4** 1/32 of the time, and device **5** 1/64 of the time, etc. Hence, although the low priority devices may suffer from long bus latency during heavy loading periods, they are not shut out from access. If the low priority devices are not especially sensitive to latency, heavy loading periods are not an issue with arbitration process **300**. If all devices (e.g., including low priority devices) are sensitive to latency, then arbitration process **300** can be modified to rotate the original priority of devices **0–5** to periodically redistribute allocation.

Referring now to FIG. **4**, an arbitration process **400** in accordance with a more versatile embodiment of the present invention is shown. Arbitration process **400** is similar to arbitration process **300**, however, arbitration process **400** provides for a more granular, finer resolution of PCI bus

bandwidth allocation and this allocation is programmable. With arbitration process **300**, as described above, device **0** receives ownership 50% of the time, device **1** 25% of the time, and so on, depending upon each device's original priority. In contrast, arbitration process **400** allows adjustable amounts of bandwidth allocation for each device by using a respective variable ownership allocation factor, N_0 through N_4, in addition to each device's original priority. Hence, arbitration process **400** provides for the adjustment of bus allocation beyond merely the priority position as in arbitration process **300**. For example, using arbitration process **400**, a system designer can allocate ownership to the highest priority device 66% of the time as opposed to 50%, or give the next highest priority device ownership 33% of the time while still giving the highest priority device ownership 50% of the time. In accordance with arbitration process **400** of the present invention, this is accomplished through modifying the allocation factors N_0 through N_4.

The allocation factors affect ownership allocation by affecting the conversion of devices from their original priority state to their interim low priority state. For example device **0**, rather than converting to the interim low priority state after a single access, waits for a predetermined number of accesses before converting to the interim low priority state. This predetermined number corresponds to the allocation factor N_0, and is programmable. By programming N_0, a customized ownership allocation is obtained. An arbiter in accordance with the present invention will allow this number of accesses to occur before treating device **0** as an interim low priority device. As with arbitration process **300**, device **0**, (as with any devices **1–5**) once in the interim low priority state, has the lowest effective priority of any of devices **1–5**. Once a lower priority device is granted ownership (e.g., any of devices **1–5**), device **0** reverts back to its original priority.

In this manner, programmable, high resolution PCI bus allocation is obtained. When one of devices **1–4** requests the PCI bus for the first time it is given its original priority. It retains this original priority for N more PCI cycles. Thereafter it converts to the interim low priority state where it is given the lowest priority until a lower priority device is granted the bus. After one of the lower priority devices is granted ownership, the interim low priority device reverts back to its original priority. Thus, arbitration process **400** avoids starvation of lower priority devices due to the fact that a high priority device is allocated ownership for a maximum of N PCI bus cycles (e.g., where a PCI bus cycle is defined as the period of time between the assertion of FRAME# to the deasserted of FRAME# and IRDY#), where N is programmable, before it converts to the interim low priority state and another device is granted ownership.

Hence, the amount of PCI bus bandwidth allocated to each device is calculated as follows:

1) For the highest priority device:
PCI bus bandwidth=$N_o/(N_o+1)$;
Where N_o is programmable from 1 to TBD (TBD is whatever maximum limit is desired) and N_o is specific to the highest priority device.

2) For lower priority devices:
PCI bus bandwidth=$N_n/(N_n+1)$ * TBR;
Where TBR is whatever the total bandwidth remaining after considering all higher priority devices. Again N_n here is programmable from 1 to TBD, where TBD is whatever maximum limit is desired for each device. There is a different N for each device.

3) For the lowest priority device:
PCI bus bandwidth=Whatever bandwidth is left over.

Thus, for example, if N is programmed to 1 for the highest priority device, 2 for the next higher priority device and 1 for each remaining device, where the total number of devices is 4, the PCI bus bandwidth allocation (for a heavily loaded bus) is as follows:

Device **0**=>PCI bus bandwidth=½ (Therefore, there is ½ the PCI bus bandwidth remaining);

Device **1**=>PCI bus bandwidth=⅔*½=⅔=⅓ (Therefore there is ⅙ the bus remaining);

Device **2**=>PCI bus bandwidth=½*⅙=1/12 (Therefore there is 1/12 the bus remaining);

Device **3**=>PCI bus bandwidth=1/12.

Referring now to FIG. **5**, a block diagram of an arbitration system **500** in accordance with one embodiment of the system of the present invention is shown. System **500** includes a plurality of PCI agents **501–507**, each coupled via a respective request grant pair (e.g., REQ **0**# GNT **0**# through REQ **6**# GNT **6**#) to a PCI arbiter **508**. Arbitration system **500**, as described above, can implement arbitration process **300** or arbitration process **400**. PCI arbiter **508** arbitrates among PCI agents **501–507** for PCI bus ownership. In the present embodiment, PCI agent **501** is the highest priority device (e.g., device **0**) and PCI agent **507** is the lowest priority device (e.g., device **6**).

Arbitration system **500**, in accordance with the present invention, ensures the lower priority devices (e.g., PCI agent **507**) are not starved of data transfer bandwidth. The interim low priority states of the present invention avoid PCI bus monopolization problems from high priority devices (e.g., PCI agent **501**). Arbitration system **500** has no potential PCI agent starvation conditions due to the fact that, in accordance with the present invention, it gives each device a chance to obtain PCI bus ownership while simultaneously preventing high priority devices from monopolizing ownership.

Thus, the present invention provides a method and system of priority arbitration which provides predictable latency and guaranteed access for devices coupled to a bus. The present invention provides an arbitration process which is much more flexible with regard to allocating bus bandwidth. The present invention ensures low priority devices are not starved while ensuring high priority devices are adequately served. In so doing, the present invention efficiently allocates bus bandwidth to maximize the overall functionality of a computer system.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the claims appended hereto and their equivalents.

What is claimed is:

1. A priority arbitration system for interfacing a plurality of peripheral component interconnect (PCI) agents coupled to a PCI bus such that high priority PCI agents are satisfied without starving low priority PCI agents, comprising:

a PCI bus adapted to transmit data signals thereon;

a high priority PCI agent coupled to said PCI bus;

a low priority PCI agent coupled to said PCI bus;

an arbiter coupled to said high priority PCI agent and said low priority PCI agent via said PCI bus;

said arbiter granting ownership of said PCI bus to said high priority PCI agent, said high priority PCI agent becoming an interim low priority PCI agent after a grant to said high priority PCI agent, said low priority PCI agent having a higher priority than said interim low priority PCI agent, said interim low priority PCI agent reverting to said high priority PCI agent subsequent to a grant to said low priority PCI agent, said arbiter granting ownership of said PCI bus to said low priority PCI agent before granting ownership of said PCI bus to said interim low priority PCI agent such that said low priority PCI agent is not prevented from accessing said PCI bus by said high priority PCI agent, and wherein said high priority PCI agent becomes said interim low priority PCI agent, said interim low priority PCI agent being the lowest priority of a plurality of PCI agents requesting ownership of said PCI bus.

2. The system of claim 1, wherein said arbiter grants ownership of said PCI bus to said high priority PCI agent, said high priority PCI agent being the highest priority of a plurality of PCI agents requesting ownership of said PCI bus.

3. The system of claim 1, wherein said high priority PCI agent becomes said interim low priority PCI agent subsequent to starting a data transaction via said PCI bus, said data transaction consequent from said grant to said high priority PCI agent from said arbiter.

4. The system of claim 1, wherein said low priority PCI agent has a higher priority than said interim low priority PCI agent such that said arbiter grants said PCI bus to said low priority PCI agent prior to said interim low priority PCI agent where said interim low priority PCI agent and said low priority PCI agent both are requesting ownership of said PCI bus.

5. The system of claim 1, wherein said interim low priority PCI agent reverts to said high priority PCI agent subsequent to said grant to said low priority PCI agent such that said high priority PCI agent again has a higher priority than said low priority PCI agent.

6. The system of claim 1, wherein a plurality of interim low priority PCI agents are prioritized with respect to each respective one of said plurality of interim low priority PCI agents.

7. A programmable priority arbitration system for interfacing a plurality of a peripheral component interconnect (PCI) agents coupled to a PCI bus such that high priority PCI agents are satisfied without starving low priority PCI agents, comprising:

a PCI bus adapted to transmit data signals thereon;

a high priority PCI agent coupled to said PCI bus;

a low priority PCI agent coupled to said PCI bus;

an arbiter coupled to said high priority PCI agent and said low priority PCI agent via said PCI bus;

said arbiter granting ownership of said PCI bus to said high priority PCI agent, said high priority PCI agent becoming an interim low priority PCI agent after a first number of transactions by said high priority PCI agent, said first number of transactions corresponding to a first allocation factor, said first allocation factor being programmable, said low priority PCI agent having a higher priority than said interim low priority PCI agent; and

said interim low priority PCI agent reverting to said high priority PCI agent subsequent to a second number of transactions by said low priority PCI agent, said second number of transactions corresponding to a second allocation factor, said second allocation factor being programmable, said arbiter granting ownership of said PCI bus to said low priority PCI agent before granting ownership of said PCI bus to said interim low priority PCI agent such that said low priority PCI agent is not prevented from accessing said PCI bus by said high priority PCI agent, and wherein said high priority PCI agent becomes said interim low priority PCI agent, said interim low priority PCI agent being the lowest priority of a plurality of PCI agents requesting ownership of said PCI bus.

8. The system of claim 7, wherein said arbiter grants ownership of said PCI bus to said high priority PCI agent, said high priority PCI agent being the highest priority of a plurality of PCI agents requesting ownership of said PCI bus.

9. The system of claim 7, wherein a plurality of interim low priority PCI agents are prioritized with respect to each respective one of said plurality of interim low priority PCI agents.

10. The system of claim 7, wherein said high priority PCI agent becomes said interim low priority PCI agent subsequent to starting said first number of data transactions corresponding to said first allocation factor, said number of data transactions consequent from at least one grant from said arbiter.

11. The system of claim 7, wherein said low priority PCI agent has a higher priority than said interim low priority PCI agent such that said arbiter grants said PCI bus to said low priority PCI agent prior to said interim low priority PCI agent, wherein said interim low priority PCI agent and said low priority PCI agent are both requesting ownership of said PCI bus.

12. The system of claim 7, wherein said interim low priority PCI agent reverts to said high priority PCI agent subsequent to said second number of transactions by said low priority PCI agent such that said high priority PCI agent again has a higher priority than said low priority PCI agent.

13. The system of claim 7, wherein a first amount of bandwidth of said PCI bus is granted to said high priority PCI agent and a second amount of bandwidth of said PCI bus is granted to said low priority PCI agent, said first amount and said second amount corresponding to said first allocation factor and said second allocation factor, wherein said first and second allocation factors are programmable to ensure respective minimum access to said PCI bus.

14. In a priority arbitration system for interfacing a plurality of a peripheral component interconnect (PCI) agents coupled to a PCI bus, a process of allocating access to the PCI bus such that high priority PCI agents are satisfied without starving low priority PCI agents, the process comprising the steps of:

a) receiving a request for ownership of a PCI bus from a first PCI agent coupled to said PCI bus;

b) receiving a request for ownership of said PCI bus from a second PCI agent coupled to said PCI bus;

c) granting ownership of said PCI bus to said first PCI agent, wherein said first PCI agent is a higher priority than said second PCI agent;

d) converting said first PCI agent to an interim low priority PCI agent such that said second PCI agent is a higher priority;

e) granting ownership of said PCI bus to said second PCI agent such that said second PCI agent is not prevented from accessing said PCI bus by said first PCI agent;

f) reverting said first PCI agent to a high priority PCI agent subsequent to granting ownership to said second PCI agent; and

g) converting said high priority PCI agent to said interim low priority PCI agent, said interim low priority PCI agent having the lowest priority of a plurality of PCI agents requesting ownership of said PCI bus.

15. The process of claim **14**, further including the step of granting ownership of said PCI bus to said high priority PCI agent, said high priority PCI agent being the highest priority PCI agent out of a plurality of PCI agents requesting ownership of said PCI bus.

16. The process of claim **14** further including the step of prioritizing a plurality of interim low priority PCI agents with respect to each respective one of said plurality of interim low priority PCI agents.

17. The system of claim **14**, further including the step of converting said high priority PCI agent to said interim low priority PCI agent subsequent to starting a data transaction via said PCI bus, said data transaction consequent from a grant to said high priority PCI agent from an arbiter.

18. The system of claim **14**, further including the step of granting said PCI bus to said low priority PCI agent prior to said interim low priority PCI agent wherein said interim low priority PCI agent and said low priority PCI agent both are requesting ownership of said PCI bus.

* * * * *